DESENVOLVIMENTO PARA

firefox OS

- GUIA RÁPIDO -



ANDRÉ GARZIA

Guia Rápido de Desenvolvimento para Firefox OS

Criando apps com HTML5 para o Firefox OS

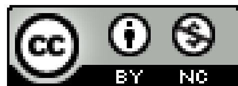
Andre Alves Garzia

This book is for sale at http://leanpub.com/guiarapidofirefoxos

This version was published on 2013-08-24



This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.



This work is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported License

Tweet This Book!

Please help Andre Alves Garzia by spreading the word about this book on Twitter!

The suggested tweet for this book is:

Estou aprendendo a desenvolver para #FirefoxOS!

The suggested hashtag for this book is #firefoxos.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#firefoxos

Para Elisangela Mendonça de Andrade Garzia

Conteúdo

Este é um livro beta

Esse livro esta sendo produzido de forma ágil. A idéia é lançar atualizações constantes melhorando e expandido o conteúdo. Como algumas APIs do Firefox OS ainda não estão finalizadas esse livro será atualizado de modo a mostrar as novidades também.

Ao contrário de alguns autores que buscam se excluir do texto vocês irão encontrar várias partes onde eu expresso minha opinião e tomo decisões que podem ir ao contrário do que outras pessoas pensam (a.k.a. vi vs emacs). Sempre que eu estiver falando da minha opinião eu deixarei isso claro no texto e espero feedback quando eu estiver errado. Quando avisado de algum erro, prometo revisar o texto!

Estou fazendo esse livro no meu tempo livre e distribuindo ele como *Creative Commons* de graça via Leanpub[1] mas escrever um livro dá bastante trabalho e eu gostaria de poder dedicar mais espaço da minha vida a esse tipo de atividade. Quem achar bacana, pode ao baixar o livro mover o preço de zero reais até quinhentos dólares e então doar algum dinheiro. Quem preferir doar algum dinheiro via PayPal, minha conta é **agarzia@mac.com**.

Quem quiser me mandar comentários e críticas pode escrever para guiarapidofxos@andregarzia.com[2]. Meu site é o http://andregarzia.com[3]. Meu Twitter é o @soapdog[4].

[1] http://leanpub.com
[2] mailto:guiarapidofxos@andregarzia.com
[3] http://andregarzia.com/pages/pt/
[4] http://twitter.com/soapdog

Ilustração da capa

Para quem gostou da ilustração da capa, ela foi feita pelo ilustrador e designer Raphael Eckhardt. Vocês podem conhecer mais do trabalho dele e entrar em contato para contratos em http://raphaeleckhardt.com/[5].

[5] http://raphaeleckhardt.com/

Para quem é esse livro

Esse livro é destinado para pessoas com conhecimentos intermediários de HTML, CSS e JavaScript que desejam construir aplicativos móveis utilizando tecnologias web. Ensinar HTML, CSS e JavaScript estão fora do escopo desse livro porém eu forneço links para bons livros de referência para quem precisar de uma forcinha.

Boas Práticas vs Facilidade de Aprendizado

Programadores experientes verão que nos códigos fontes apresentados nesse livro nem sempre as melhores práticas para programação estão sendo seguidas. Eu não estou encapsulando meu JavaScript em funções imediatas. Eu estou colocando coisas que não devia no objeto global. Isso tudo tem um motivo, facilitar o entendimento e a leitura do código. Esse livro é um guia introdutório e programadores experientes saberão onde e como modificar o código. Programadores novatos entenderão o que está escrito e não sofrerão nenhum problema pois tudo que está aqui funciona.

Se você deseja aprofundar seus estudos de JavaScript e se tornar o **mestre ninja king size da web** eu vou passar uma lista de livros abaixo.

- JavaScript: The Good Parts[6]: O livro de JavaScript.
- JavaScript Patterns[7]: Um livro sobre padrões de programação em JavaScript.
- JavaScript Enlightenment[8]: Técnicas avançadas para JavaScript.
- Maintainable JavaScript[9]: Escrevendo códigos fáceis de manter.

Vários desses livros também estão disponíveis em Português. Outras opções boas são os livros da **Casa do Código** que são super atualizados e escritos por autores nacionais.

- A Web Mobile[10]
- HTML5 e CSS 3[11]

---

[6] http://shop.oreilly.com/product/9780596517748.do

[7] http://shop.oreilly.com/product/9780596806767.do

[8]

[9] http://shop.oreilly.com/product/0636920027713.do

[10] http://www.casadocodigo.com.br/products/livro-web-mobile

[11] http://www.casadocodigo.com.br/products/livro-html-css

Feedback & Pull Requests

Esse livro é livre e eu aguardo todo tipo de *feedback* que vocês possam me dar. Todo o conteúdo do livro está em um repositório no GitHub[12] e quem tiver recomendações, bug fixes ou melhorias pode enviar um *pull request* sem medo. Agradeço desde já todas as contribuições.

O repositório do livro é https://github.com/soapdog/guia-rapido-firefox-os[13].

---

[12]https://github.com/soapdog/guia-rapido-firefox-os

[13]https://github.com/soapdog/guia-rapido-firefox-os

Histórico

Versão 0.1

Esse é a primeira versão desse livro. Ele não passou na mão de nenhum editor e nem foi revisto. O que você está lendo aqui começou a ser escrito no dia 20 de Agosto de 2013 para ser distribuído durante a BrazilJS dias 22 e 23 de Agosto portanto você esta lendo uma versão bem crua porém eu estou partindo do princípio que é melhor ter a informação agora.

Estou utilizando o sistema da Leanpub[14] para edição desse livro. Esse sistema ainda tem uns problemas de localização em partes do template do livro e erros de unicode no índice. Eu já enviei um patch de localização para a empresa e eles já estão cientes do problema do unicode no índice e prometeram resolver em breve (a.k.a. uma semana). Então dentro dessas limitações atuais do sistema peço desculpas por qualquer inconveniente causado por esses bugs, eles serão solucionados em breve.

[14]http://leanpub.com

Introdução

O Firefox OS



Firefox OS

O Firefox OS[15] é a nova plataforma móvel livre desenvolvida pela Mozilla[16] e pelos seus parceiros. Aparelhos com Firefox OS já estão à venda em diversos países e serão lançados ainda esse ano no Brasil.

Voltado inicialmente para mercados emergentes, o Firefox OS tem como objetivo trazer o próximo milhão de pessoas para a web. Para conseguir isso os aparelhos com Firefox OS são construídos para serem ótimos como *o primeiro smartphone* de alguém e tem preços muito competitivos. No Brasil, 78% das pessoas ainda estão usando *feature phones*, o objetivo é ser uma alternativa interessante para essas pessoas migrarem desses aparelhos para smartphones rodando Firefox OS.

Infelizmente em mercados emergentes como o Brasil smartphones com uma performance aceitável ainda são muito caros. Pessoas podem comprar smartphones baratos mas as plataformas utilizadas atualmente para esse tipo de aparelho estão sendo construídas com foco em

[15] http://www.mozilla.org/pt-BR/firefox/os/
[16] http://mozilla.org

smartphones de alta performance deixando esses aparelhos baratos com uma performance ruim e indesejável.

Outro fator importante quando falamos de Firefox OS é que os atuais sistemas móveis mais populares são pequenas ilhas proprietárias onde você está amarrado as vontades do fabricante que possui privilégios de mandar e desmandar na plataforma. Nesses sistemas proprietários você em geral só pode distribuir seus aplicativos em canais autorizados e o fabricante fica com um percentual de todas as transações financeiras que passam pelo aplicativo.

Além de amarrar os desenvolvedores através das lojinhas de apps, esses sistemas possuem sistemas de desenvolvimento próprios incompatíveis entre si como por exemplo a Apple com o Objective-C/Cocoa e a Google com o Java para Android. Dessa forma, para construir um app nativo para iOS e Android, o desenvolvedor precisa aprender as duas linguagens e recriar o programa duas vezes. O Firefox OS traz para o mundo mobile uma proposta diferente ao ter o HTML5 como sistema de desenvolvimento de apps nativos. O HTML5 é o sistema aberto e livre utilizado pelos navegadores modernos da web e aplicativos construídos baseados em tecnologia possuem potencial para serem multiplataforma com facilidade (da menos trabalho para garantir que um app web funciona em várias plataformas do que construir o mesmo app várias vezes para cada uma).

A plataforma que o HTML5 merece

A web está em todo lugar desde o seu computador, ao seu telefone celular, à sua SmartTV e até no seu videogame. A linguagem da programação da web, o JavaScript, é uma das linguagens mais difundidas no mundo estando presente em basicamente todos os tipos de aparelhos[17]. Quando as pessoas falam sobre HTML5 elas estão em geral falando da união de três tecnologias: O HTML 5, CSS 3 e o JavaScript. Essa tríade é super poderosa, o HTML 5 simplifica o HTML e expande suas capacidades em relação ao XHTML 1.0, o CSS 3 vem com mais capacidades para layout e animação e o JavaScript de hoje é uma linguagem fantástica que serve tanto para iniciantes quanto para programadores experientes.

O Firefox OS é basicamente uma extensão móvel da web onde o HTML5 está em primeiro lugar. Ao tornar o HTML5 um cidadão de primeira classe em sua plataforma a Mozilla torna o novo sistema acessível à milhões de desenvolvedores web. O HTML5 funciona muito bem em navegadores modernos em desktops e laptops porém não havia antes do Firefox OS uma plataforma móvel que fizesse jus ao mesmo. Enquanto outros vendedores produzem navegadores que implementam o HTML5, o Firefox OS vai além disso implementando não só o HTML5 mas também toda uma série de APIs para acesso ao hardware via JavaScript.

Acesso ao hardware com WebAPI

Outras plataformas anteriores também tentaram criar sistemas operacionais cuja criação de aplicativos estava baseada em tecnologias web. Assim que o iPhone foi lançado, a única maneira de criar apps era através de webapps. O WebOS também utilizava HTML, CSS e JavaScript para

[17]Para ter uma idéia assista a palestra JavaScript Anywhere do Jaysdon.

a criação de apps. O que diferencia o Firefox OS dessas plataformas é que ele oferece acesso ao hardware e a componentes do sistema via JavaScript. No iOS os webapps não tem esse tipo de acesso e portanto apps baseados *apenas em web* se tornam cidadãos de segunda classe incapazes de competir com aplicativos nativos.

Por acesso ao hardware estamos falando de coisas como por exemplo acessar os contatos do telefone, enviar SMS, acessar a câmera e as fotos do aparelho. No Firefox OS graças a coleção de APIs chamadas de WebAPI[18], o desenvolvedor pode aproveitar todas essas funcionalidades utilizando nada além das tecnologias do HTML5.

Outra diferença é que ao contrário de plataformas anteriores como por exemplo o WebOS que também promovia o acesso ao hardware via JavaScript porém que não tentou tornar suas APIs um padrão da web a ser adotado pelos outros vendedores, a Mozilla está trabalhando em conjunto com o W3C e outros grupos para que a WebAPI se torne um padrão aberto da web e possa ser implementado por outros vendedores como por exemplo o Google e a Apple. Conforme as APIs forem implementadas pelos demais fabricantes, seus aplicativos precisarão de cada vez menos mudanças para funcionar em plataformas diferentes.

A WebAPI não esta sendo implementada somente para o Firefox OS, a Mozilla também esta implementando a mesma API no Firefox para Desktop e no Firefox Mobile no Android. Assim aplicativos construídos com base nessas APIs estarão aptos a rodar no Desktop, no Firefox OS e no Android, onde quer que o Firefox rode.

Liberdade para desenvolvimento e distribuição

Como tudo da Mozilla, o Firefox OS é construído as claras e livre. Todo o desenvolvimento pode ser acompanhado pelo GitHub da Mozilla[19]. Com o Firefox OS você tem a liberdade para acompanhar e contribuir com o desenvolvimento do sistema e também a liberdade para distribuir seus aplicativos como desejar seja na sua própria página web ou no Firefox Marketplace[20].

A idéia principal é que você não fique preso a Mozilla para nada. Se você quiser pegar o código fonte do sistema e modifica-lo para as suas necessidades você pode. Se quiser construir aplicativos para utilização interna da sua empresa ou se quiser distribuir suas criações somente em sua própria página você também pode. Em outras plataformas, você esta em geral amarrado a distribuir seus aplicativos somente via a loja autorizada do fabricante e portanto sujeito aos critérios e ao processo de aprovação do mesmo. O Firefox Marketplace também possui um processo e critérios de aprovação porém você é livre para não utilizar ele se não quiser. Assim como na web, onde você é livre para hospedar sua página como achar melhor, no Firefox OS você também é.

Conclusão

Em resumo o HTML 5 chegou para ficar e melhora a cada dia que passa. O Firefox OS que é o novo sistema operacional móvel da Mozilla totalmente livre e construído as claras oferece

[18]https://wiki.mozilla.org/WebAPI
[19]https://github.com/mozilla-b2g/B2G
[20]https://marketplace.firefox.com/

uma implementação robusta do HTML 5 e vai além ao oferecer APIs de acesso ao hardware via JavaScript. Essas APIs estão sendo padronizadas junto aos órgãos competentes e promovidas para adoção por outros fabricantes.

No próximo capítulo vamos ver o que é necessário para criar aplicativos para Firefox OS. Vamos juntos que em breve já teremos um app rodando.

Se Preparando Para Desenvolver Apps Para Firefox OS

Para construir e testar apps feitos para Firefox OS você não precisa de nada além de uma versão recente do Firefox, do complemento chamado Firefox OS Simulator e um bom editor de texto para programação[21].

Obtendo o Firefox

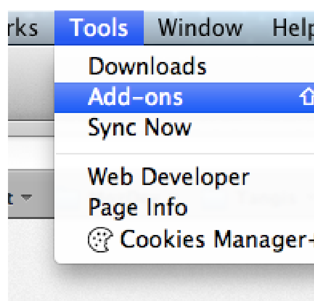
Diferentes navegadores possuem diferentes motores de renderização e execução de JavaScript. O Google Chrome, o Opera e o Safari utilizam o motor conhecido como WebKit (ou Blink que é um fork do WebKit), o Firefox utiliza o motor chamado Gecko tanto no desktop como no Android e no Firefox OS portanto para desenvolver apps para Firefox OS é melhor utilizar o Firefox para Desktop pois ambos utilizam o mesmo sistema de renderização e execução de JavaScript.

Além disso, a Mozilla disponibiliza um simulador do Firefox OS que funciona como um complemento do Firefox portanto, é necessário instalar o Firefox para desktop para poder rodar o simulador do Firefox OS.

A versão estável atual do Firefox pode ser obtida nessa página[22] e então basta seguir as instruções para instalar no seu sistema operacional preferido.

Instalando o Simulador do Firefox OS

Após a instalação do Firefox, o próximo passo é a instalação do simulador do Firefox OS que será utilizado para testarmos nossos aplicativos. Com o Firefox instalado e rodando, vá no menu **Ferramentas** e selecione **Complementos**[23] como pode ser visto na imagem abaixo:



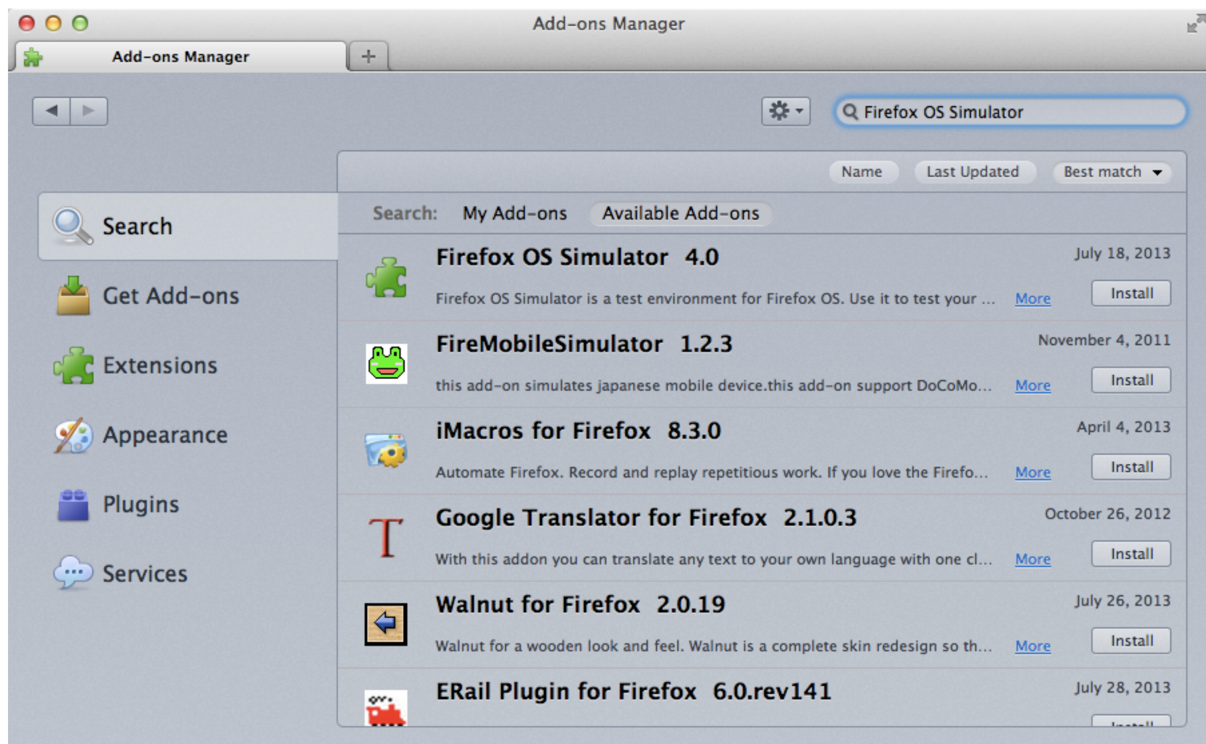
Menu Ferramentas com menu Complementos selecionado

[21]Existem vários editores bons com diferentes graus de complexidade e funcionalidades. Um muito popular que eu recomendo para quem ainda não possui um editor favorito é o SublimeText. Pessoalmente eu utilizo o WebStorm que é uma IDE completa para criação de web apps.

[22]http://getfirefox.com

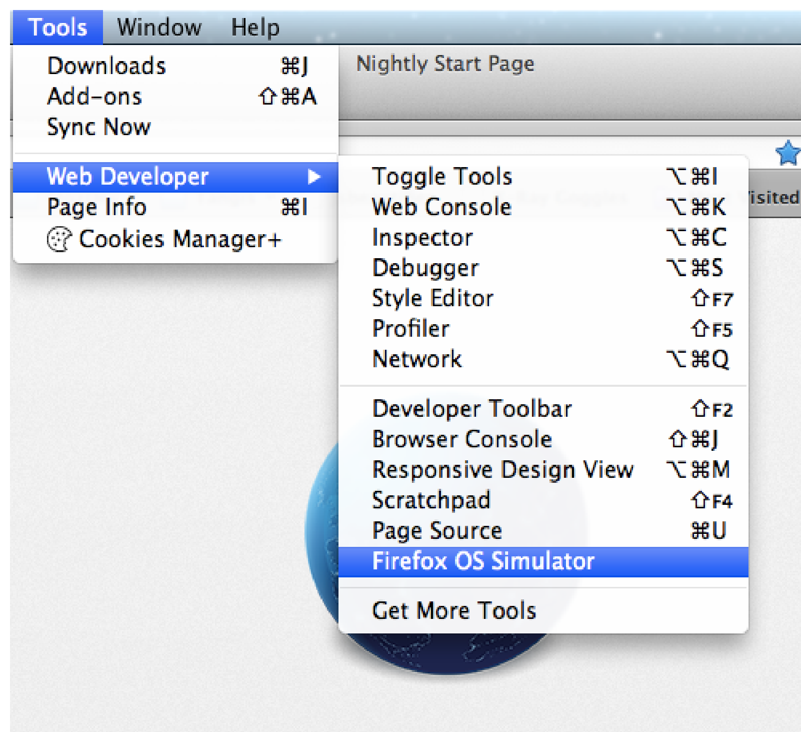
[23]Caso seu sistema esteja em ingles procure o menu **Tools** e então **Add-ons**.

Utilizando a ferramenta de busca presente no canto superior direito, procure por **firefox os simulator** e instale o simulador.



Gerenciador de complementos mostrando o simulador

Após o complemento ser instalado, ele estará disponível no menu **Ferramentas -> Desenvolvedor Web -> Simulador do Firefox OS**



Onde fica o simulador após instalado

Conclusão

Agora que temos todos o ambiente preparado vamos aprender uns conceitos básicos para então construirmos nosso primeiro app.

Conceitos

Antes de colocarmos as mãos na massa e construírmos nosso primeiro app vamos aprender alguns conceitos básicos para desenvolvimento para Firefox OS. Como sabemos os apps são baseados em HTML5 assim como uma página web mas ainda não sabemos o que diferencia um app de uma simples página web. Se pensarmos as demais plataformas móveis que conhecemos podemos chegar a alguns requisitos mínimos para um aplicativo móvel:

- Possuir um ícone.
- Possuir um nome.
- Funcionar offline quando possível.

No Firefox OS, um aplicativo é a grosso modo uma página web que possui um ícone, um nome e pode funcionar offline dependendo de como o desenvolvedor fizer o seu trabalho. Todas essas informações a respeito de um aplicativo tais como nome, ícone entre outras são definidas em um arquivo de manifesto como veremos na próxima sessão.

O Manifesto

O manifesto[24] é um arquivo do tipo JSON[25] que descreve um aplicativo. Em geral esse arquivo se chama **manifest.webapp** e fica ao lado do seu arquivo principal HTML que normalmente chama **index.html**.

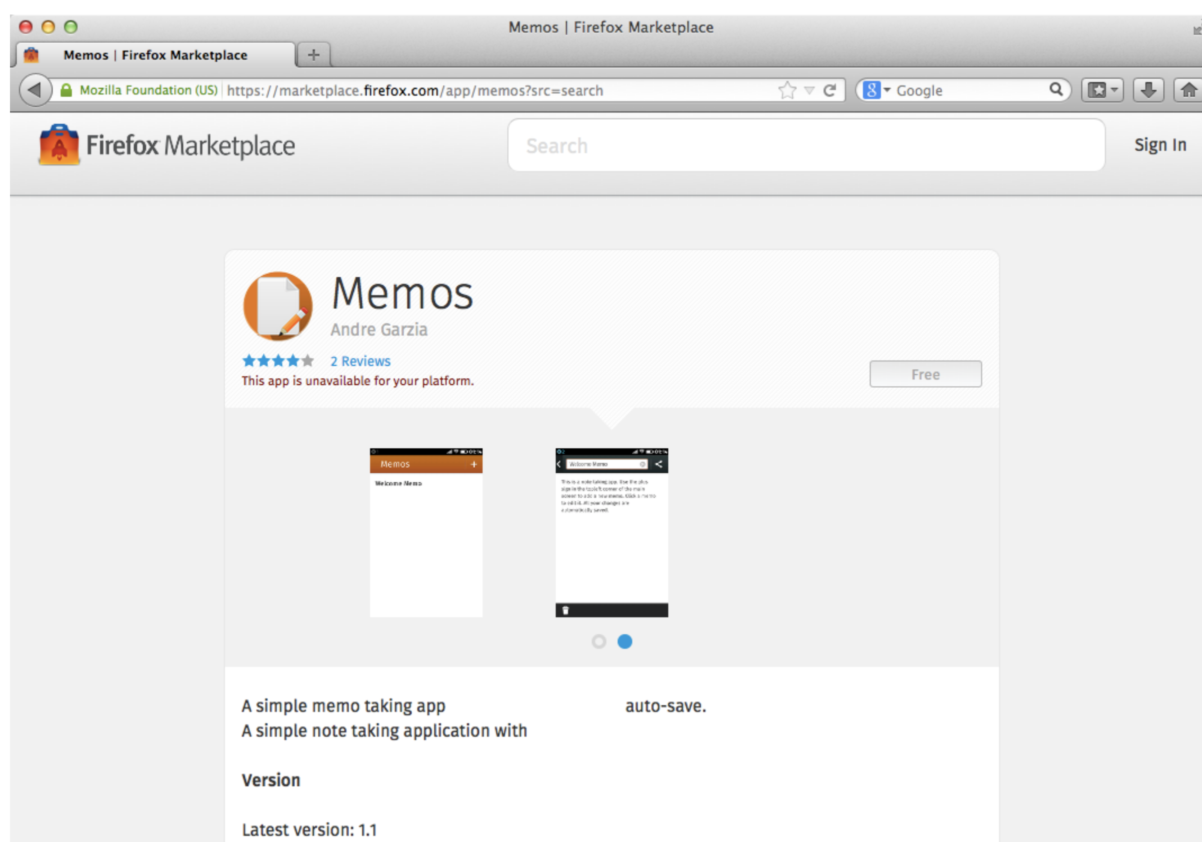
Exemplo de Manifesto

```
{
  "name": "Memos",
  "version": "1.1",
  "description": "A simple memo taking app",
  "launch_path": "/index.html",
  "permissions": {
    "storage": {
      "description": "Required for storing and retrieving notes."
      }
  },
  "developer": {
    "name": "Andre Garzia",
    "url": "http://andregarzia.com"
```

[24] https://developer.mozilla.org/pt-BR/docs/Apps/Manifest
[25] http://json.org

```
  },
  "icons": {
    "60": "/style/icons/icon_60.png",
    "128": "/style/icons/icon_128.png"
  }
}
```

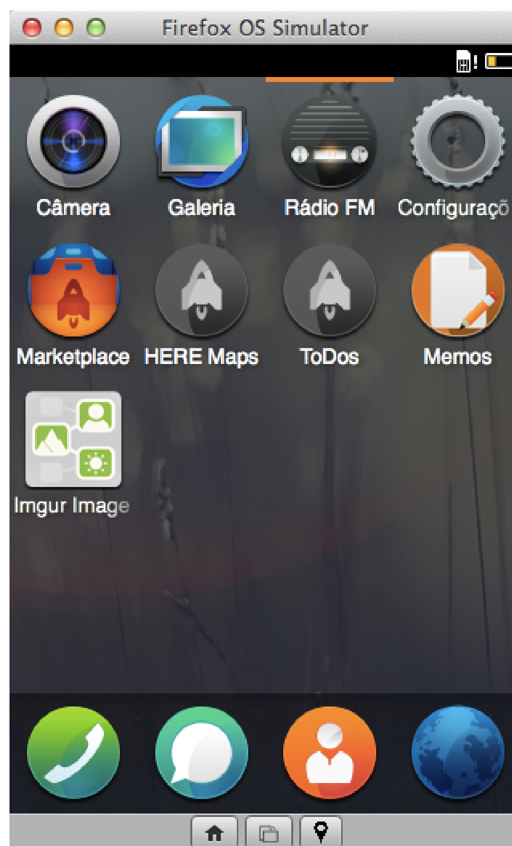
Acima podemos ver um exemplo de manifesto do aplicativo chamado memos[26]. Entre outras coisas ele descreve quem fez o aplicativo, qual icone e nome do mesmo, qual arquivo é utilizado para carregar o app (nesse caso o index.html), quais permissões de acesso ao hardware ele precisa, etc. Esse arquivo é utilizado pelo Firefox OS para adicionar o aplicativo ao aparelho e pelo Firefox Marketplace para criar a listagem do mesmo na loja como podemos ver na imagem abaixo:



Listagem do memos no Firefox Marketplace

A mesma informação do manifesto é utilizada para colocar o aplicativo no aparelho como podemos ver na captura de tela do simulador a seguir:

[26] Esse é um app de exemplo para Firefox OS que pode ser visto no Firefox Marketplace e cujo o código fonte esta disponível no github



Memos no simulador

De posse dos seus arquivos HTML, CSS, JavaScript e de um arquivo de manifesto, você ja tem um app pronto para rodar no Firefox OS. Continuando sobre o tópico sobre aplicativos vamos aprender sobre os tipos de aplicativos existentes

Tipos de aplicativo

No Firefox OS existem dois tipos de aplicativos: aplicativos hospedados e aplicativos empacotados.

- **Aplicativos Hospedados**: ficam armazenados em um servidor web assim como uma site e quando utilizados pelo usuário é feito um acesso ao servidor remoto caso o app não esteja no cache.
- **Aplicativos Empacotados**: são distribuídos como um arquivo zip e são copiados para o aparelho durante a instalação.

Existem prós e contras para as duas soluções. Aplicativos hospedados são mais fáceis de atualizar pois basta trocar os arquivos no servidor porém são mais difíceis de fazer funcionar offline pois precisam da utilização do **appcache**[27] para isso. Outro ponto é que mesmo utilizando o appcache, o aplicativo demora um pouco mais para ligar pois o cache precisa ser verificado. Aplicativos

[27]https://developer.mozilla.org/pt-BR/docs/HTML/Using_the_application_cache

empacotados por outro lado estão no telefone desde a instalação e não precisam verificar cache nem nada antes de executar porém a atualização é mais complexa e incluí o envio da nova versão do app para o Firefox Marketplace e/ou a criação de rotinas de atualização dentro do mesmo caso você distribua o app fora do marketplace.

Não existe uma regra sobre qual tipo de aplicativo usar. Pessoalmente, eu prefiro construir aplicativos empacotados para não ter que lidar com o appcache e não ter que ficar hospedando o app em algum servidor. Aplicativos empacotados distribuídos no Firefox Marketplace ficam hospedados nos servidores da Mozilla e para mim isso é mais fácil. Vale a pena dizer que o mundo das ferramentas para desenvolvimento com JavaScript ja produziu geradores de arquivos de appcache que facilitam muito a vida de quem esta criando apps hospedados. A escolha é de cada um porém durante esse livro utilizarei aplicativos empacotados pois é mais fácil e mantém nosso foco nos apps e não em hospedar e servir apps via web. Para quem quiser saber mais sobre aplicativos hospedados, basta checar o link de aplicativos hospedados na central do desenvolvedor[28].

Níveis de acesso ao hardware

Existem três níveis de acesso ao hardware no Firefox OS e cada nível possui APIs que pode e que não pode acessar.

- **Normal**: Os aplicativos normais possuem acesso as WebAPIs mais frequentemente utilizadas tais como geolocalização, pegar foto da câmera. Aplicativos hospedados e aplicativos empacotados que não declarem um tipo no manifesto são por definição normais.
- **Privilegiado**: Um aplicativo privilegiado tem acesso a todas as APIs disponíveis para um app normal e mais algumas. Uma exigência é que todos os aplicativos privilegiados sejam empacotados, ou seja, você não pode ter um aplicativo hospedado que seja privilegiado. Esses aplicativos tem acesso a APIs mais "profundas" do Firefox OS como por exemplo API de contatos sem interação com usuário.
- **Certificado**: Aplicativos certificados tem acesso total ao hardware e só podem ser construídos pela Mozilla e seus parceiros de hardware. Eles tem acesso por exemplo ao sistema de telefonia. Um exemplo de aplicativo certificado é o discador do Firefox OS.

Em geral a maioria dos aplicativos não precisa de nada além do que o acesso normal oferece porém as vezes é necessário um acesso privilegiado para podermos utilizar certas APIs. Quando criamos um aplicativo privilegiado e enviamos ele ao Firefox Marketplace o processo de aprovação é mais rigoroso (e isso é bom).

Na página sobre as WebAPIs no wiki da Mozilla[29] podemos ver quais APIs estão implementadas em quais plataformas e qual o nível de acesso necessário para utiliza-las.

[28]https://marketplace.firefox.com/developers/docs/hosted
[29]https://wiki.mozilla.org/WebAPI

Geolocation API	Access to the end user's location.	W3C CR	D	A	B	Security
WiFi Information API	Privileged API to get a list of available WiFi networks. Also get signal strength and name of currently connected network, etc.	Future?	D	A	B	Security

Acesso normal

Acesso privilegiado

Níveis de acesso para cada API

Como podemos ver na imagem acima, o acesso a *Geolocation API* está disponível para todos os aplicativos enquanto a API *WIFI Information API* está disponível somente para aplicativos privilegiados.

As WebAPIs

No Firefox OS não precisamos de nada além das tecnologias web para construir apps tão capazes quanto os apps nativos das outras plataformas. Toda a parte de acesso ao hardware é realizada através das WebAPIs. Para conhecer a lista de APIs disponíveis para a versão atual do Firefox OS basta visitar a página sobre as WebAPIs no wiki da Mozilla[30].

Para ilustrar quão fáceis são essas APIs vou mostrar alguns exemplos de uso a seguir.

Exemplo #1: Realizar ligações

Imaginemos que você tem um programa que precisa abrir o discador do telefone com um número preenchido. Basta utilizar o código abaixo:

Enviando um número para o telefone

```
var call = new MozActivity({
  name: "dial",
  data: {
    number: "5555-9999"
  }
});
```

Esse código apenas abre o discador com o número preenchido, portanto o usuário do telefone precisa apertar o botão de discar para efetuar a ligação. Esse tipo de API que precisa de uma ação do usuário antes de executar sua função é bem comum e é uma forma de segurança afinal utilizando essa API você não tem como criar um programa que liga para algum lugar sem interferência do usuário. Outras APIs que são capazes de efetuar ligações sem a confirmacão do usuário estão disponíveis para níveis mais elevados de aplicativos. A API do exemplo está disponível para todos os aplicativos.

[30] https://wiki.mozilla.org/WebAPI

Essa API é uma Web Activity, para saber mais sobre esse tipo de API visite este artigo no blog da Mozilla[31].

Exemplo #2: Salvar um contato

Salvando um contato

```
var contact = new mozContact();
contact.init({name: "Odin"});

var request = navigator.mozContacts.save(contact);
request.onsuccess = function() {
  // contato salvo com sucesso
};
request.onerror = function() {
  // não foi possível salvar o contato
};
```

Essa API cria um objeto com os dados do contato e salva ele para a agenda do telefone sem interferência do usuário e está disponível somente para aplicativos privilegiados. Esse padrão onde se cria um objeto com um callback de sucesso e um de erro é muito utilizado nas WebAPIs.

Para saber mais sobre a API de contatos, visite a pagina da *Contacts API* no wiki da Mozilla[32].

Exemplo #3: Pegando uma imagem da camera

Pegando uma imagem

```
var getphoto = new MozActivity({
  name: "pick",
  data: {
    type: ["image/png", "image/jpg", "image/jpeg"]
  }
});

getphoto.onsuccess = function () {
  var img = document.createElement("img");
  if (this.result.blob.type.indexOf("image") != -1) {
    img.src = window.URL.createObjectURL(this.result.blob);
  }
};
```

[31]https://hacks.mozilla.org/2013/01/introducing-web-activities/
[32]https://wiki.mozilla.org/WebAPI/ContactsAPI

```

getphoto.onerror = function () {
        // erro!
};
```

Aqui vemos mais um exemplo de uma WebActivity[33] (falaremos de web activities mais adiante no livro). As Web Activities estão disponíveis para todos os aplicativos. No caso desse exemplo utilizamos uma atividade do tipo *pick* e passamos os *MIME Types* desejados. Ao executar esse código, o sistema mostra uma tela para o usuário pedindo para ele selecionar da onde a imagem deve vir (câmera, álbum, wallpapers) e caso o usuário selecione uma imagem a callback de sucesso é executada, caso ele cancele a ação a callback de erro é executada. No caso de sucesso, o programa recebe um blob com a imagem. Na imagem abaixo podemos ver um exemplo do workflow:



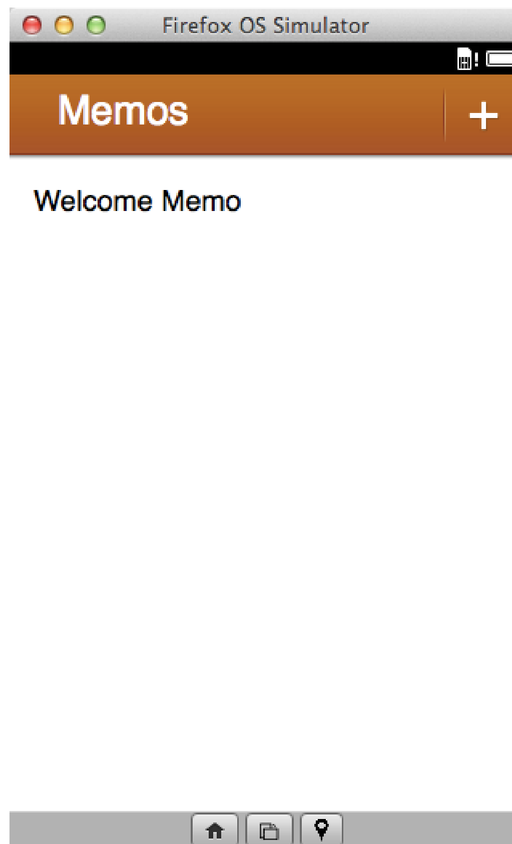
Exemplo da atividade *pick*

Conclusão

Nesse capítulo vimos alguns conceitos básicos sobre o Firefox OS que serão expandidos nos próximos capítulos. Agora é hora de colocar a mão na massa e fazermos um app!

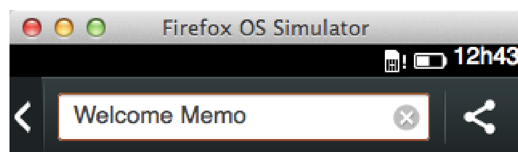
[33]https://hacks.mozilla.org/2013/01/introducing-web-activities/

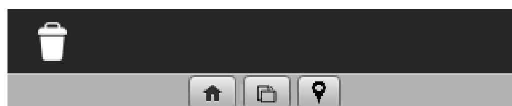
Nosso Primeiro App



Memos, um bloco de notas minimalista

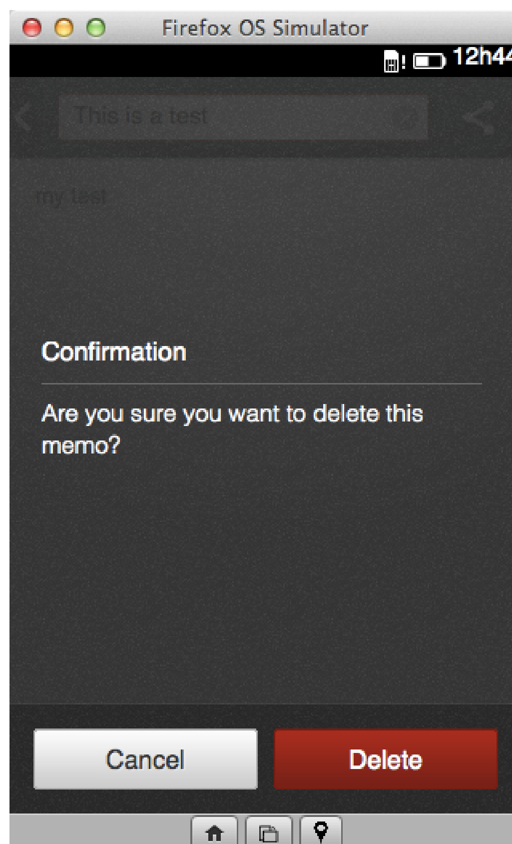
Nesse capítulo vamos reconstruir o aplicativo **Memos** que é um bloco de notas que eu criei para servir de exemplo em minhas palestras. Esse programa possui três telas. A primeira está acima e é a tela principal do programa que lista os títulos das notas. Ao clicar em uma dessas notas ou clicar no sinal de adição somos direcionados para a tela de edição onde podemos alterar o título e conteúdo da nota como pode ser visto abaixo:





Memos, tela de edição

Nesta tela de edição o usuário pode também deletar a nota sendo mostrada ao clicar no ícone da lixeira e confirmar o desejo de deletar a nota como a captura de tela mostrada a seguir.



Memos, confirmar a remoção de uma nota

O código do Memos está disponível no meu Github[34] para quem quiser baixar e olhar logo tudo de uma vez. Existe uma cópia do código do app na pasta **code** dentro do repositório do livro[35] que contém o código fonte em Markdown utilizado para gerar esse livro.

O Memos utiliza IndexedDB[36] para armazenar as notas e o Gaia Building Blocks[37] para a construção da sua interface. Em uma atualização futura deste livro eu falarei mais sobre os building blocks, nesta primeira versão simplesmente construiremos o app.

O primeiro passo é separarmos uma pasta para o aplicativo. Vamos chamar a pasta de **memos**.

Criando o manifesto

O manifesto do Memos é bem simples. Crie um arquivo chamado **manifest.webapp** no a pasta **memos**. Manifestos são arquivos do tipo JSON[38] que descrevem um aplicativo nele colocamos o nome, a descrição, os ícones utilizados e muitas outras coisas importantes como quais permissões o programa necessita para funcionar e qual arquivo é utilizado para carregar o app.

Abaixo podemos ver o conteúdo do manifesto do Memos, atenção ao copiar pois é fácil errar uma vírgula e tornar o seu JSON inválido. Para validar o seu JSON você pode utilizar várias ferramen-

[34] https://github.com/soapdog/memos-for-firefoxos
[35] https://github.com/soapdog/guia-rapido-firefox-os
[36] https://developer.mozilla.org/en-US/docs/IndexedDB/Using_IndexedDB
[37] http://buildingfirefoxos.com/building-blocks
[38] http://json.org

tas, uma delas que é especifica para validação de manifestos que é o http://appmanifest.org/[39]. Para aprender mais sobre manifestos visite a página na MDN sobre manifestos[40].

Manifesto do programa Memos (*manifest.webapp*)

```
{
  "name": "Memos",
  "version": "1.1",
  "description": "A simple memo taking app",
  "launch_path": "/index.html",
  "permissions": {
    "storage": {
      "description": "Required for storing and retrieving notes."
      }
  },
  "developer": {
    "name": "Andre Garzia",
    "url": "http://andregarzia.com"
  },
  "icons": {
    "60": "/style/icons/icon_60.png",
    "128": "/style/icons/icon_128.png"
  }
}
```

Vamos explicar cada um dos campos do manifesto acima.

Campo	Descrição
name	Esse campo dita o nome do aplicativo.
version	Essa é a versão atual do seu aplicativo. Mudar a versão causa um update no app.
launch_path	Qual arquivo que deve ser carregado quando o seu app é iniciado
permissions	Quais permissões seu app precisa. Mais informações sobre permissões adiante
developer	Quem desenvolveu esse aplicativo
icons	Os ícones para cada tamanho necessário

A parte mais interessante desse manifesto é a entrada de permissões onde pedimos a permissão para *storage* que permite que utilizemos o IndexedDB sem limitação de espaço[41] (graças a isso podemos armazenar quantas notas quisermos no nosso programa).

Com o manifesto pronto podemos passar para o HTML.

[39]http://appmanifest.org/
[40]https://developer.mozilla.org/pt-BR/docs/Apps/Manifest
[41]Para saber mais sobre as permissões que você pode pedir olhe a página na MDN sobre permissões de aplicativos.

Estruturando o HTML

Antes de colocarmos a mão na massa e montarmos o HTML utilizado pelo memos vamo falar rapidamente sobre o Gaia Building Blocks[42] que é uma iniciativa de construir um conjunto de css e js reutilizáveis com o *look and feel* do Firefox OS para você aproveitar nos seus próprios apps.

No Firefox OS, assim como na web em geral, você não é obrigado a utilizar o *look and feel* do Firefox OS. Utilizar ou não os Building Blocks é uma decisão sua que passa por questões de *branding*, conveniência de uso, adequação ao que você precisa entre outras, o importante é entender que você não sofre nenhum tipo de repreensão no Firefox Marketplace por não utilizar a cara do Firefox OS. Eu como não sou um bom designer opto sempre por utilizar um pacote pronto como esse (ou contratar um designer).

A estrutura do HTML do nosso programa foi construída para se adequar aos padrões adotados pelo Gaia Building Blocks onde cada tela é uma *section* e os elementos seguem uma formulazinha. O ideal agora é que você baixe o código fonte do Memos a partir do meu Github[43] para que você tenha os arquivos do Building Blocks.



Aviso: A versão que eu usei do Building Blocks não é a atual e eu modifiquei alguns arquivos portanto o código que vamos mostrar só vai funcionar com a versão que está no repositório do Memos.

Incluíndo os Building Blocks

Antes de mais nada, copie as pastas **shared** e **style** para a pasta Memos para que possamos utilizar o Building Blocks. Vamos começar o nosso arquivo **index.html** com os *includes* necessários para o programa.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="/style/base.css" />
    <link rel="stylesheet" type="text/css" href="/style/ui.css" />
    <link rel="stylesheet" type="text/css" href="/style/building_blocks.css" />
    <link rel="stylesheet" type="text/css" href="shared/style/headers.css" />
    <link rel="stylesheet" type="text/css" href="shared/style_unstable/lists.c\
ss" />
    <link rel="stylesheet" type="text/css" href="shared/style_unstable/toolbar\
s.css" />
```

[42] http://buildingfirefoxos.com/building-blocks
[43] https://github.com/soapdog/memos-for-firefoxos

```
13      <link rel="stylesheet" type="text/css" href="shared/style/input_areas.css"\
14  />
15      <link rel="stylesheet" type="text/css" href="shared/style/confirm.css" />
16      <title>Memos</title>
17  </head>
```

Na *linha 01* declaramos o tipo do documento como sendo HTML 5. Da *linha 05 até a linha 15* incluímos os CSS dos diversos componentes que são utilizados no aplicativo tais como cabeçalhos, listas, áreas de entrada de dados entre outros.

Construíndo a tela principal

Agora podemos passar a implementação das telas. Como falamos anteriormente, cada tela do programa é uma **<section>** dentro do **<body>** do HTML que deve ter um atributo *role* com valor *application* tipo `<body role="application">`. Isso é utilizado pelos seletores dos CSS do Building Blocks. Vamos construír a primeira tela (e declarar o body).

```
<body role="application">

<section role="region" id="memo-list">
    <header>
        <menu type="toolbar">
            <a id="new-memo" href="#"><span class="icon icon-add">add</span></\
a>
        </menu>
        <h1>Memos</h1>
    </header>
    <article id="memoList" data-type="list"></article>
</section>
```

Nossa tela tem um **<header>** que possui um botão para adicionar novas notas e o nome do programa. Possui também um **<article>** que é utilizado para conter a lista de notas armazanadas no app. Nós utilizaremos as IDs do **<article>** e do **botão** para capturar eventos quando chegarmos na parte em JavaScript.

Repare que a criação da tela é um HTML bem tranquilo de se entender, construir a mesma tela em outras linguagens é muito mais trabalhoso. Simplesmente declaramos nossos componentes e damos IDs para elementos que desejamos referenciar posteriormente.

Agora que temos a tela principal pronta, vamos montar a tela de edição que é mais complicada.

Montando a tela de edição

```
<section role="region" id="memo-detail" class="skin-dark hidden">
    <header>
        <button id="back-to-list"><span class="icon icon-back">back</span>
        </button>
        <menu type="toolbar">
            <a id="share-memo" href="#"><span class="icon icon-share">share</s\
pan>
            </a>
        </menu>
        <form action="#">
            <input id="memo-title" placeholder="Memo Title" required="required\
" type="text">
            <button type="reset">Remove text</button>
        </form>
    </header>
    <p id="memo-area">
        <textarea placeholder="Memo content" id="memo-content"></textarea>
    </p>
    <div role="toolbar">
        <ul>
            <li>
                <button id="delete-memo" class="icon-delete">Delete</button>
            </li>
        </ul>
    </div>
    <form id="delete-memo-dialog" role="dialog" data-type="confirm" class="hid\
den">
        <section>
            <h1>Confirmation</h1>
            <p>Are you sure you want to delete this memo?</p>
        </section>
        <menu>
            <button id="cancel-delete-action">Cancel</button>
            <button id="confirm-delete-action" class="danger">Delete</button>
        </menu>
    </form>
</section>
```

Essa tela de edição contém a tela de diálogo utilizada quando o usuário tenta deletar uma nota por isso ela é mais complicada.

No topo da tela que é marcado pelo **<header>** temos o botão de voltar para a tela principal, uma caixa de entrada de texto que é utilizada para mostrar e modificar o título da nota e um botão utilizado para enviar a nota por email.

Depois da toolbar que fica no topo, temos um parágrafo contendo uma área para a entrada de texto da nota e então uma outra toolbar com um botão para deletar a nota que está aberta.

Esses três elementos e seus nós filhos formam a tela de edição e após essa tela temos um **<form>** que na verdade representa a caixa de diálogo utilizada pela tela de confirmação da remoção da nota. Essa caixa de diálogo é bem simples contendo uma mensagem informativa e um botão para cancelar a ação de deletar a nota e um para confirmar.

Ao fecharmos essa **<section>** terminamos todas as telas do programa e o restante do código HTML serve apenas para incluir os arquivos de JavaScript utilizados pelo programa.

```
<script src="/js/model.js"></script>
<script src="/js/app.js"></script>
</body>
</html>
```

Construíndo o JavaScript

Agora vamos programar de verdade e dar vida ao nosso app. Para efeitos de organização separei o código em dois arquivos de JavaScript:

- **model.js**: que contém as rotinas para lidar com o armazenamento e alteração das notas porém não contém a lógica do programa ou algo relacionado a sua interface e tratamento de entrada de dados.
- **app.js**: responsável por ligar os elementos do HTML às rotinas correspondentes e contém a lógica do app.

Os dois arquivos devem ser postos em uma pasta chamada **js** ao lado das pastas **style** e **shared**.

model.js

No Firefox OS utilizaremos o IndexedDB[44] para guardar as notas. Como pedimos a permissão de *storage* podemos grava quantas notas a memória do aparelho permitir.

A parte do código do model.js que mostrarei abaixo é responsável por abrir a conexão e criar o *storage* se necessário.

> Importante: Esse código foi escrito para ser entendido facilmente e não representa as melhores práticas de programação para JavaScript. Variáveis globais são utilizadas (ARGH!) entre outros problemas. Fora isso o tratamento de erros é basicamente inexistente. O mais importante desse livro é ensinar o *worlflow* de como programar apps para Firefox OS.

[44] https://developer.mozilla.org/en-US/docs/IndexedDB/Using_IndexedDB

```
var dbName = "memos";
var dbVersion = 1;

var db;
var request = indexedDB.open(dbName, dbVersion);

request.onerror = function (event) {
    console.error("Can't open indexedDB!!!", event);
};
request.onsuccess = function (event) {
    console.log("Database opened ok");
    db = event.target.result;
};

request.onupgradeneeded = function (event) {

    console.log("Running onUpgradeNeeded");

    db = event.target.result;

    if (!db.objectStoreNames.contains("memos")) {

        console.log("Creating objectStore for memos");

        var objectStore = db.createObjectStore("memos", {
            keyPath: "id",
            autoIncrement: true
        });
        objectStore.createIndex("title", "title", {
            unique: false
        });

        console.log("Adding sample memo");
        var sampleMemo1 = new Memo();
        sampleMemo1.title = "Welcome Memo";
        sampleMemo1.content = "This is a note taking app. Use the plus sign in\
 the topleft corner of the main screen to add a new memo. Click a memo to edit\
 it. All your changes are automatically saved.";

        objectStore.add(sampleMemo1);
    }
}
```

> Importante: Novamente me perdoem pelas variáveis globais, isso aqui é um app educativo apenas. Outro detalhe é que eu removi os comentários dos código colados no livro para economizar espaço. O código fonte no github está comentado.

O código acima cria um objeto *db* e um objeto *request*. O objeto *db* é utilizado por outras funções no código para manipular o registro das notas.

Na implementação da função `request.onupgradeneeded` aproveitamos para criar uma nota de exemplo desta forma assim que o programa liga pela primeira vez e essa função é executada, o banco de dados é inicializado com uma nota de boas vindas.

Com nossa conexão aberta e armazenamento inicializado é hora de implementar as funções para manipulação das notas.

```
function Memo() {
    this.title = "Untitled Memo";
    this.content = "";
    this.created = Date.now();
    this.modified = Date.now();
}

function listAllMemoTitles(inCallback) {
    var objectStore = db.transaction("memos").objectStore("memos");
    console.log("Listing memos...");

    objectStore.openCursor().onsuccess = function (event) {
        var cursor = event.target.result;
        if (cursor) {
            console.log("Found memo #" + cursor.value.id + " - " + cursor.valu\
e.title);
            inCallback(null, cursor.value);
            cursor.continue();
        }
    };
}

function saveMemo(inMemo, inCallback) {
    var transaction = db.transaction(["memos"], "readwrite");
    console.log("Saving memo");

    transaction.oncomplete = function (event) {
        console.log("All done");
    };

```

```
    transaction.onerror = function (event) {
        console.error("Error saving memo:", event);
        inCallback({
            error: event
        }, null);

    };

    var objectStore = transaction.objectStore("memos");

    inMemo.modified = Date.now();

    var request = objectStore.put(inMemo);
    request.onsuccess = function (event) {
        console.log("Memo saved with id: " + request.result);
        inCallback(null, request.result);

    };
}

function deleteMemo(inId, inCallback) {
    console.log("Deleting memo...");
    var request = db.transaction(["memos"], "readwrite").objectStore("memos").\
delete(inId);

    request.onsuccess = function (event) {
        console.log("Memo deleted!");
        inCallback();
    };
}
```

Acima criamos uma função construtora para montar novas notas já com alguns campos inicializados. A seguir implementamos funções para listar, salvar e remover as notas. Essas funções em geral sempre aceitam um parametro `inCallback` que é uma função de retorno para ser executada após o processamento da função. Isso é necessário dada a natureza assíncrona das chamadas ao IndexedDB. Todas as callbacks tem a mesma assinatura que é `callback(error, value)` onde um dos valores é nulo dependendo do que aconteceu.

> Como esse é um livro de caráter introdutório eu optei por não utilizar *Promises*[a] visto que muitos desenvolvedores ainda não estão familiarizados com o conceito. Eu recomendo fortemente a utilização desse tipo de solução que torna o código mais legível e fácil de manter.
>
> [a] https://developer.mozilla.org/en-US/docs/Mozilla/JavaScript_code_modules/Promise.jsm/Promise

Com nossas funções para o armazenamento e manipulação de notas prontas vamos agora implementar a lógica da nossa app no arquivo **app.js**

app.js

Esse arquivo contém a nossa lógica do programa. Como o código é grande demais para colocar todo de uma vez só vou quebra-lo em partes e colocar aos pouquinhos.

```
var listView, detailView, currentMemo, deleteMemoDialog;

function showMemoDetail(inMemo) {
    currentMemo = inMemo;
    displayMemo();
    listView.classList.add("hidden");
    detailView.classList.remove("hidden");
}


function displayMemo() {
    document.getElementById("memo-title").value = currentMemo.title;
    document.getElementById("memo-content").value = currentMemo.content;
}

function shareMemo() {
    var shareActivity = new MozActivity({
        name: "new",
        data: {
            type: "mail",
            body: currentMemo.content,
            url: "mailto:?body=" + encodeURIComponent(currentMemo.content) + "\
&subject=" + encodeURIComponent(currentMemo.title)

        }
    });
    shareActivity.onerror = function (e) {
        console.log("can't share memo", e);
    };
}

function textChanged(e) {
    currentMemo.title = document.getElementById("memo-title").value;
    currentMemo.content = document.getElementById("memo-content").value;
    saveMemo(currentMemo, function (err, succ) {
        console.log("save memo callback ", err, succ);
        if (!err) {
            currentMemo.id = succ;
```

```
        }
    });
}

function newMemo() {
    var theMemo = new Memo();
    showMemoDetail(theMemo);
}
```

Primeiro declaramos algumas variáveis globais para armazenar umas referências a alguns elementos que pretendemos utilizar dentro das funções. Das variáveis globais, a mais interessante é `currentMemo` que é um objeto que guarda qual nota o usuário está vendo.

As funções `showMemoDetail()` e `displayMemo()` funcionam juntas, a primeira carrega a nota escolhida em `currentMemo` e manipula o CSS dos elementos para que a tela de visualização/edição seja mostrada e a segunda se encarrega de pegar o conteúdo de `currentMemo` e adicionar a tela. Dava para fazer as duas coisas dentro de uma função só mas quando eu estava experimentando com esse app eu separei as funções para que a inserção do conteúdo da nota na interface ficasse separado da manipulação das telas, desta forma eu pude brincar mais com variações das funções.

A função `shareMemo()` utiliza uma WebActivity[45] para criar uma nova mensagem no programa de email com o conteúdo da nota.

A função `textChanged()` pega os dados dos campos de entrada e coloca novamente em `currentMemo` e então salva a nota. Isso é feito pois a idéia do programa é que o usuário nunca precise explicitamente salvar uma nota. Toda alteração nas caixas de entrada de texto causam a execução dessa rotina e a atualização da nota no banco de dados.

A função `newMemo()` cria uma nova nota e abre a tela de edição para ela.

```
function requestDeleteConfirmation() {
    deleteMemoDialog.classList.remove("hidden");
}

function closeDeleteMemoDialog() {
    deleteMemoDialog.classList.add("hidden");
}

function deleteCurrentMemo() {
    closeDeleteMemoDialog();
    deleteMemo(currentMemo.id, function (err, succ) {
        console.log("callback from delete", err, succ);
        if (!err) {
            showMemoList();
        }
    });
```

[45] https://hacks.mozilla.org/2013/01/introducing-web-activities/

```
}

function showMemoList() {
    currentMemo = null;
    refreshMemoList();
    listView.classList.remove("hidden");
    detailView.classList.add("hidden");
}
```

A função `requestDeleteConfirmation()` é responsável por mostrar a tela de confirmação de remoção de nota.

As funções `closeDeleteMemoDialog()` e `deleteCurrentMemo()` são acionadas pelos botões de cancelamento e de confirmação de remoção da nota.

A função `showMemoList()` faz a faxina antes de mostrar a lista de notas pois limpa o valor de `currentMemo` afinal se estamos vendo a lista de notas então não estamos vendo nenhuma nota e faz com que a tela principal seja mostrada.

```
function refreshMemoList() {
    if (!db) {
        // HACK:
        // this condition may happen upon first time use when the
        // indexDB storage is under creation and refreshMemoList()
        // is called. Simply waiting for a bit longer before trying again
        // will make it work.
        console.warn("Database is not ready yet");
        setTimeout(refreshMemoList, 1000);
        return;
    }
    console.log("Refreshing memo list");

    var memoListContainer = document.getElementById("memoList");


    while (memoListContainer.hasChildNodes()) {
        memoListContainer.removeChild(memoListContainer.lastChild);
    }

    var memoList = document.createElement("ul");
    memoListContainer.appendChild(memoList);

    listAllMemoTitles(function (err, value) {
        var memoItem = document.createElement("li");
        var memoP = document.createElement("p");
        var memoTitle = document.createTextNode(value.title);

```

```
        memoItem.addEventListener("click", function (e) {
            console.log("clicked memo #" + value.id);
            showMemoDetail(value);

        });

        memoP.appendChild(memoTitle);
        memoItem.appendChild(memoP);
        memoList.appendChild(memoItem);


    });
}
```

A função `refreshMemoList()` modifica o DOM ao construir elemento por elemento a lista de notas contidas no banco de dados. Seria muito mais fácil utilizar uma template com handlebars ou coisa semelhante mas como esse é um app *vanilla javascript* fizemos tudo na mão. Essa função é chamada pela `showMemoList()` que foi mostrada anteriormente.

Essas são todas as funções do nosso programa. A única parte de código que falta é a inicialização dos eventos e a chamada inicial para `refreshMemoList()`.

```
window.onload = function () {
    // elements that we're going to reuse in the code
    listView = document.getElementById("memo-list");
    detailView = document.getElementById("memo-detail");
    deleteMemoDialog = document.getElementById("delete-memo-dialog");

    // All the listeners for the interface buttons and for the input changes
    document.getElementById("back-to-list").addEventListener("click", showMemo\
List);
    document.getElementById("new-memo").addEventListener("click", newMemo);
    document.getElementById("share-memo").addEventListener("click", shareMemo);
    document.getElementById("delete-memo").addEventListener("click", requestDe\
leteConfirmation);
    document.getElementById("confirm-delete-action").addEventListener("click",\
 deleteCurrentMemo);
    document.getElementById("cancel-delete-action").addEventListener("click", \
closeDeleteMemoDialog);
    document.getElementById("memo-content").addEventListener("input", textChan\
ged);
    document.getElementById("memo-title").addEventListener("input", textChange\
d);

    // the entry point for the app is the following command
    refreshMemoList();
```

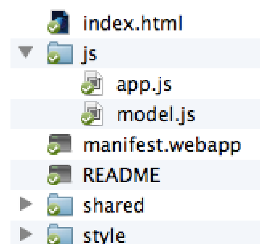
```

};
```

Agora todos os nossos arquivos estão prontos e podemos testar o aplicativo no simulador.

Testando o app no simulador

Antes de testarmos o aplicativo no simulador é melhor garantirmos que todos os arquivos estão no lugar certo. Sua pasta deve se parecer com essa aqui:

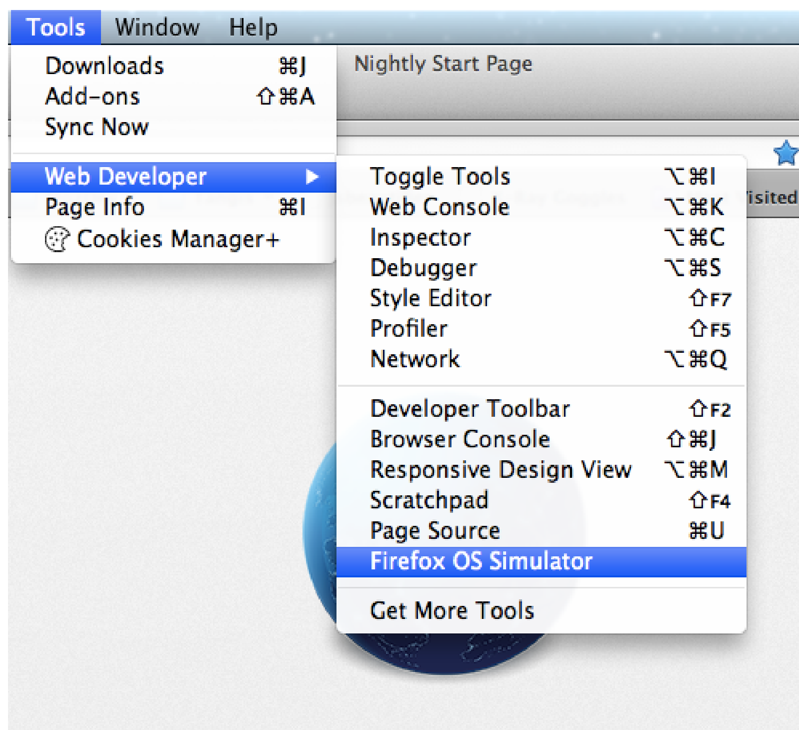


Lista de arquivos do Memos

Se você desconfia que pode haver algo errado com o que você digitou basta comparar sua versão com a que está no meu GitHub[46] (Existe uma cópia do código do app na pasta **code** dentro do repositório do livro[47]).

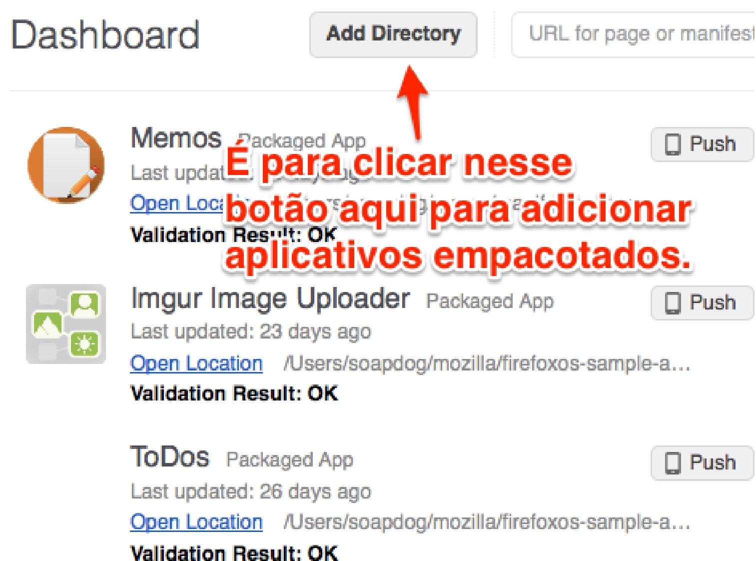
Para abrir o *Dashboard do Simulator* vá no menu **Ferramentas** -> **Desenvolvedor Web** -> **Firefox OS Simulator**.

[46] https://github.com/soapdog/memos-for-firefoxos
[47] https://github.com/soapdog/guia-rapido-firefox-os



Como abrir o dashboard do simulador

Com o dashboard aberto, clique no botão **Add Directory** e navegue até a pasta do código fonte do Memos e selecione o arquivo de manifesto.



Adicionando um novo aplicativo

Se tudo funcionar corretamente você vera o aplicativo Memos na lista de apps.



Memos aparecendo no dashboard

Ao adicionar o app, o simulador ligará automaticamente com o app aberto para você testar. Pronto, você ja pode testar as funcionalidades do Memos. Parabéns você criou e testou seu primeiro app. Não é um app complexo ou revolucionário mas te ajudou a entender o *workflow* de desenvolvimento. Lembre-se sempre que você alterar o código fonte do aplicativo, você deve apertar o botão **Refresh** para atualizar a cópia que está instalada no simulador.

Conclusão

Parabéns você construiu seu primeiro aplicativo móvel para Firefox OS e testou ele no simulador. No próximo capítulo apresentaremos as ferramentas do desenvolvedor que irão facilitar a sua vida na hora de programar.

Ferramentas do Desenvolvedor

O Firefox possui diversas ferramentas para auxiliar os desenvolvedores web a fazerem o seu trabalho. Muita gente ainda usa o FireBug[48] e não sabe que o Firefox ja inclui ferramentas próprias. Nessa sessão vamos ver rapidamente algumas ferramentas que são muito úteis para quem está desenvolvendo apps.

Quem estiver interessado em conhecer mais sobre essas ferramentas e o que mais virá por ai pode dar uma olhada na página da MDN sobre ferramentas do desenvolvedor[49].

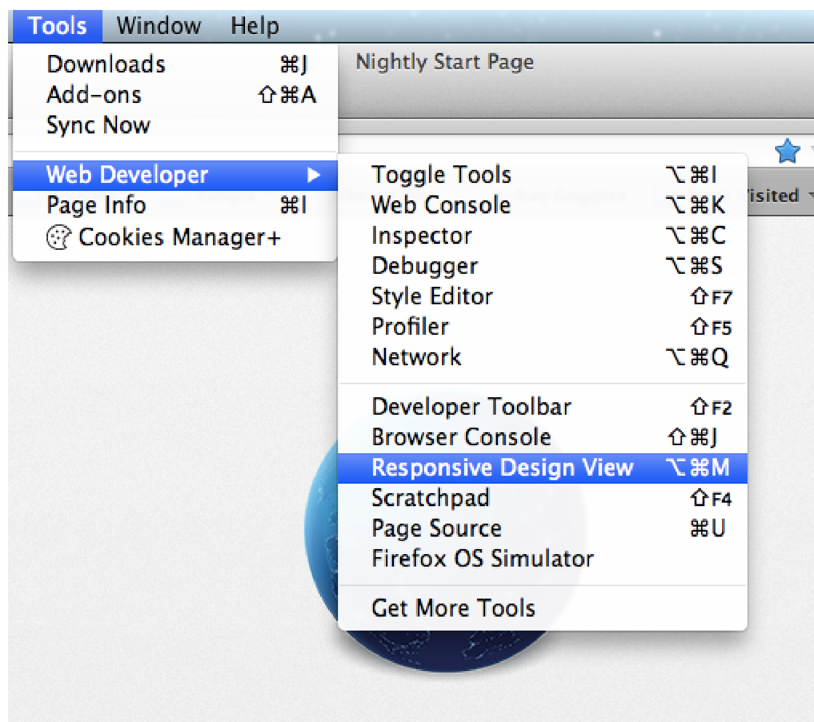
Conhecendo o Modo de Design Adaptável

Uma das coisas mais cômodas de quando estamos desenvolvendo para web é o fato de podermos simplesmente salvar o HTML e recarregar no browser para vermos as mudanças sem a necessidade de um compilador ou coisa do gênero. Por mais que o simulador do Firefox OS também permita esse tipo de workflow as vezes queremos simplesmente ir testando as coisas no Firefox no desktop mesmo. Esse tipo de teste no desktop é muito comum quando estamos lidando com aplicativos hospedados que devem se adaptar para desktops, tablets e telefones. Nesses casos você pode utilizar o **Modo de Design Adaptável** para adaptar a tela (e o viewport) para tamanhos comuns de tablet e telefones e ver como seu trabalho fica nessas telas.

A utilização do modo de design adaptável é especialmente importante para quem está trabalhando com **media queries**[50] pois permite que você redimensione a tela do jeito que bem entender e teste o seu CSS.

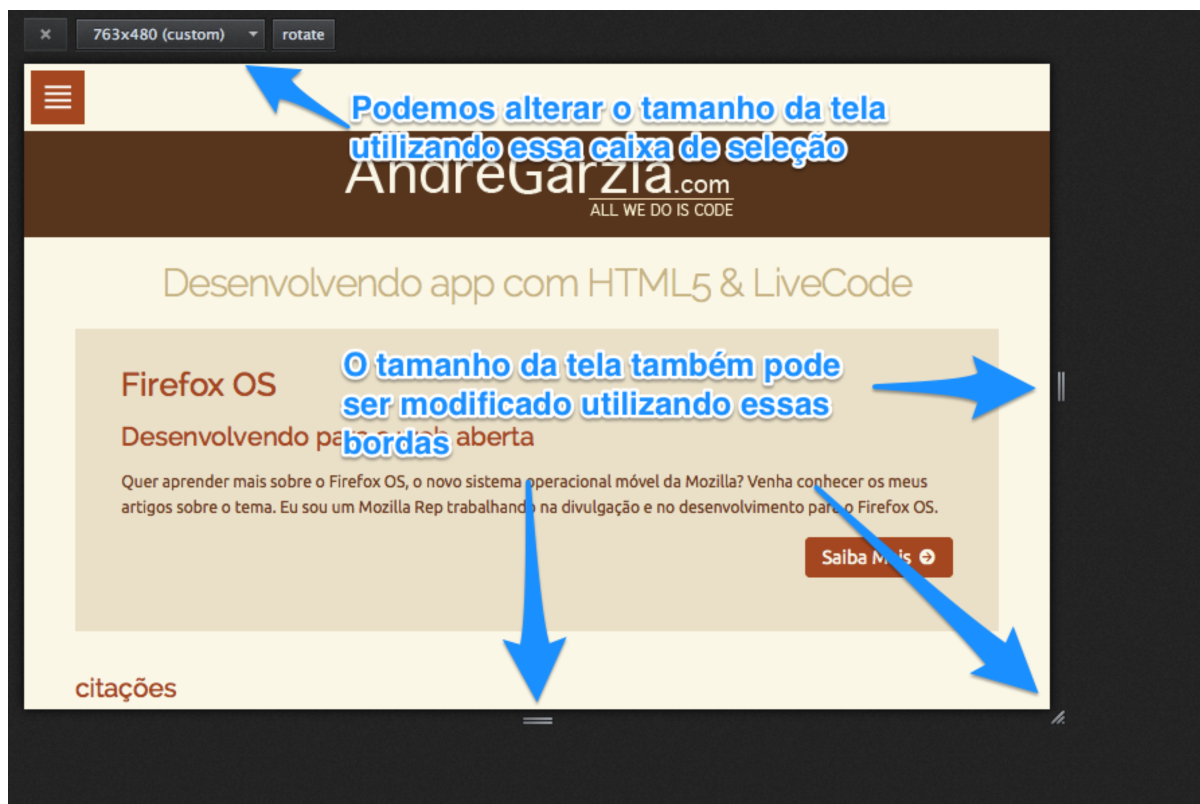
O modo de design adaptável pode ser ativado no **menu Ferramentas** -> **Desenvolvedor Web** -> **Modo de Design Adaptável**[51] como podemos ver na imagem abaixo.

[48]https://addons.mozilla.org/pt-BR/firefox/addon/firebug/
[49]https://developer.mozilla.org/en-US/docs/Tools
[50]https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries
[51]Em inglês *Responsive Design View*.



Ativando o Modo de Design Adaptável

Ao ativar esse modo, a tela do Firefox se modifica de maneira que você possa alterar o tamanho do viewport utilizando as laterais da tela ou a caixa de seleção.



Exemplo do Modo de Design Adaptável

A maioria dos telefones rodando Firefox OS que já estão no mercado funcionam com uma resolução de 480x320 porém em vez de simplesmente fazer as coisas pensando nessa resolução é mais legal utilizar media queries e a metodologia chamada de design responsivo. Para quem quer saber mais sobre design responsivo eu recomendo o livro Responsive Web Design[52] e o Mobile First[53]. A editora nacional **Casa do Código**[54] esta com livros muito bons a preços muito acessíveis também, dos livros do catálogo deles, os mais interessantes para o assunto em questão são Web Design Responsivo[55] e A Web Mobile[56]. Para quem quiser entrar de cabeça, eles tem um pacote de livros de web responsiva e mobile[57].

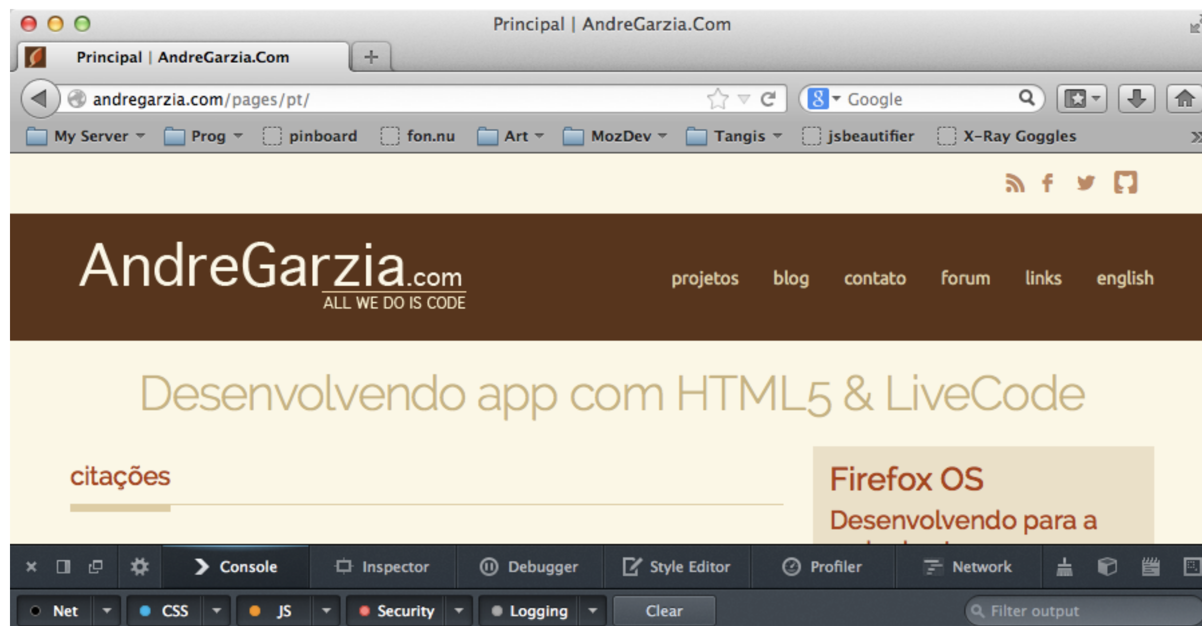
Em resumo o modo de design adaptável permite que a gente teste os nossos apps em diversos tamanhos de tela sem ser necessário ficar redimensionando a janela principal do Firefox. Na minha opinião é uma das ferramentas mais úteis do mundo e eu não sei por que os demais navegadores ainda não copiaram…

Outras Ferramentas

As ferramentas de desenvolvedor do Firefox são similares às do FireBug e do Google Chrome. Elas permitem que você interaja com o DOM e com o ambiente do JavaScript da página que está

[52]http://www.abookapart.com/products/responsive-web-design
[53]http://www.abookapart.com/products/mobile-first
[54]http://casadocodigo.com.br
[55]http://www.casadocodigo.com.br/products/livro-web-design-responsivo
[56]http://www.casadocodigo.com.br/products/livro-web-mobile
[57]http://www.casadocodigo.com.br/products/colecao-webdesign

carregada e muito mais. Com elas você pode testar suas funções JavaScript, enviar comandos de depuração e status via o objeto console[58] e manipular tanto o DOM como o CSS da página. São ferramentas imprescindíveis para quem está trabalhando com web.



Página com o Console de JavaScript visível

Além do *console de JavaScript* existem várias outras ferramentas como um *editor de estilo*[59], *monitor de rede*[60], *profiler de JavaScript*[61], *depurador de JavaScript*[62], *inspetor de páginas*[63] entre outros.

Como vimos no aplicativo que construímos no capítulo anterior, o console pode ser utilizado para verificar o andamento do nosso programa. Existem muitos desenvolvedores web que ainda utilizam *alerts()* espalhados pelo código para debugar coisas, aprender a utilizar as ferramentas do desenvolvedor é um passo muito importante na formação de qualquer um (sem contar que poupa um tempo desgraçado).

Uma ferramenta que vale um destaque é o *depurador remoto*[64] que permite que você conecte um telefone rodando Android ou Firefox OS ao seu computador e utilize as ferramentas de

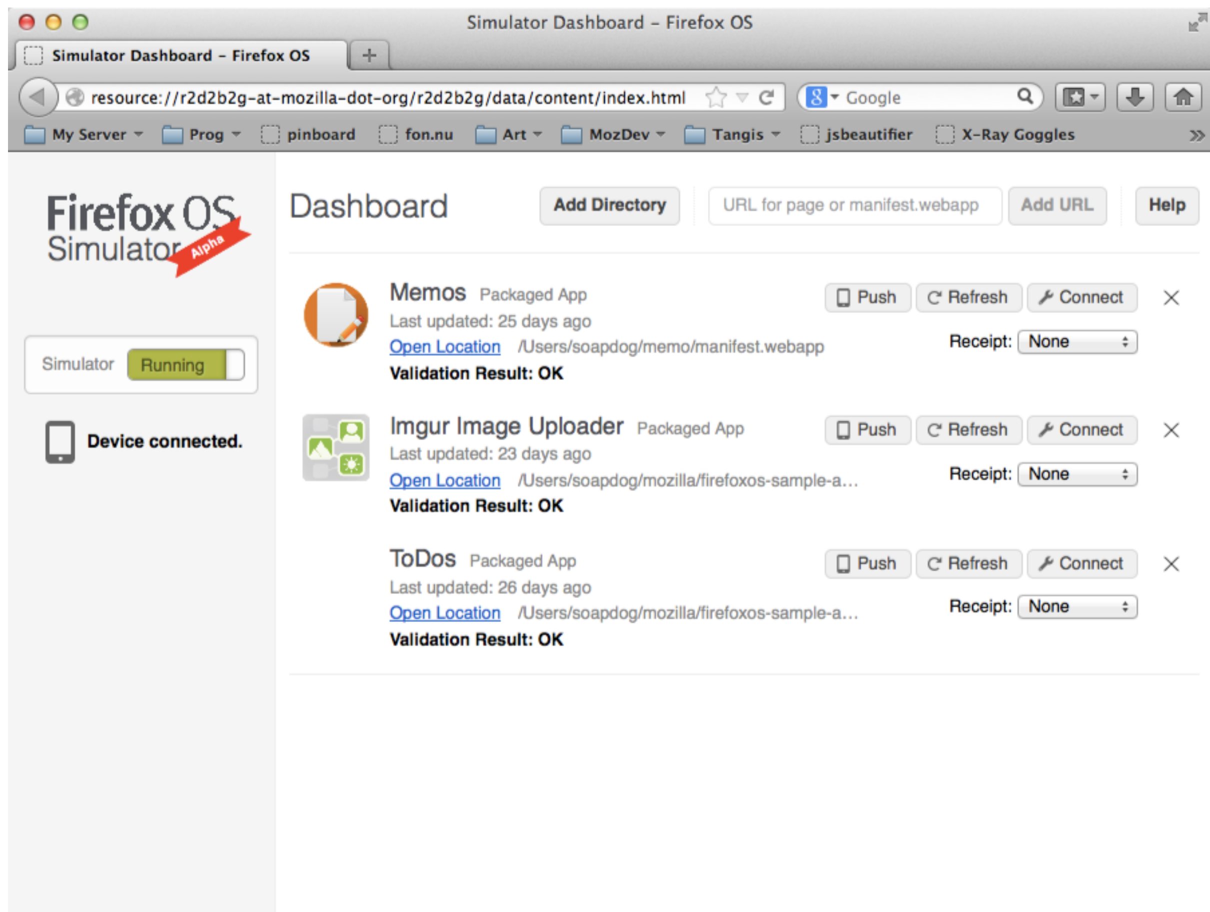
[58]https://developer.mozilla.org/en-US/docs/Web/API/console
[59]https://developer.mozilla.org/en-US/docs/Tools/Style_Editor
[60]https://developer.mozilla.org/en-US/docs/Tools/Network_Monitor
[61]https://developer.mozilla.org/en-US/docs/Tools/Profiler
[62]https://developer.mozilla.org/en-US/docs/Tools/Debugger
[63]https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector
[64]https://developer.mozilla.org/en-US/docs/Tools/Remote_Debugging

desenvolvedor do seu computador para depurar o que esta rodando no aparelho móvel.

Conclusão

Esse foi um capítulo expositivo para dar ao leitor referências para aprender mais sobre as ferramentas que vem dentro do Firefox. A utilização dessas ferramentas facilita muito a vida da gente e quando aliamos elas com o simulador do Firefox OS temos uma combinação imbatível para construção de apps por isso que no próximo capítulo vamos aprender um pouco mais sobre o simulador.

O Simulador do Firefox OS



Dashboard do Simulador do Firefox OS

No capítulo Se Preparando Para Desenvolver Apps Para Firefox OS nós instalamos o simulador do Firefox OS, no capítulo Nosso Primeiro App utilizamos o simulador para testar o memos, agora vamos aprofundar o nosso conhecimento sobre o mesmo vendo como as tarefas mais comuns são realizadas.

Para saber mais sobre o simulador visite a página na MDN sobre o simulador[65].

[65]https://developer.mozilla.org/en-US/docs/Tools/Firefox_OS_Simulator

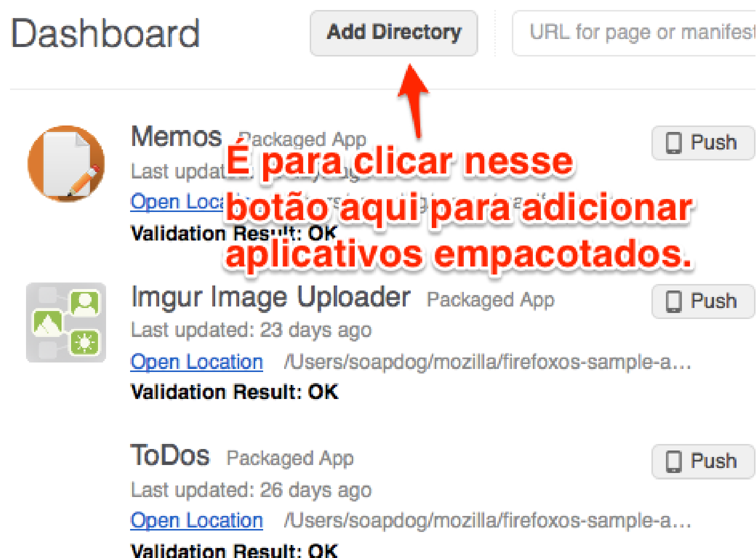
Adicionando Apps

Você pode adicionar aplicativos empacotados ou hospedados no simulador. O procedimento é similar porém com diferenças sutis por isso vamos ver cada um deles separadamente.

Adicionando aplicativos empacotados

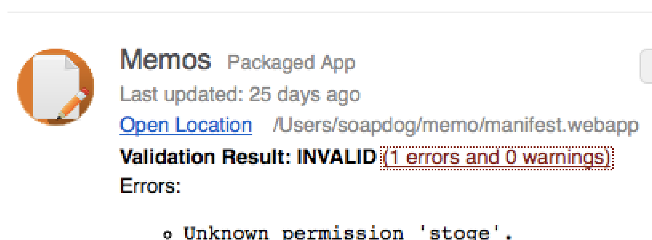
O foco desse livro são os aplicativos empacotados simplesmente por que eu acho mais fácil trabalhar com eles portanto você ja sabe como fazer para adicionar um app que está em uma pasta dentro do seu computador afinal nós fizemos isso quando fomos testar o Nosso Primeiro App porém vamos repetir a explicação aqui para que fique mais claro.

Para adicionar um aplicativo empacotado devemos clicar no botão **Add Directory** no **Dashboard do Simulador** como podemos ver abaixo.



Mostrando o botão utilizado para adicionar aplicativos empacotados ao simulador

Ao clicar no botão destacado na imagem, o Firefox abre uma janela para seleção de arquivos. Nessa janela você deve navegar no seu HD e localizar o **arquivo de manifesto** do aplicativo que deseja adicionar. Se tudo ocorrer bem, o app sera adicionado e o simulador ligará com o app em execução. Caso tenha algo de errado com o seu manifesto o simulador irá avisar.



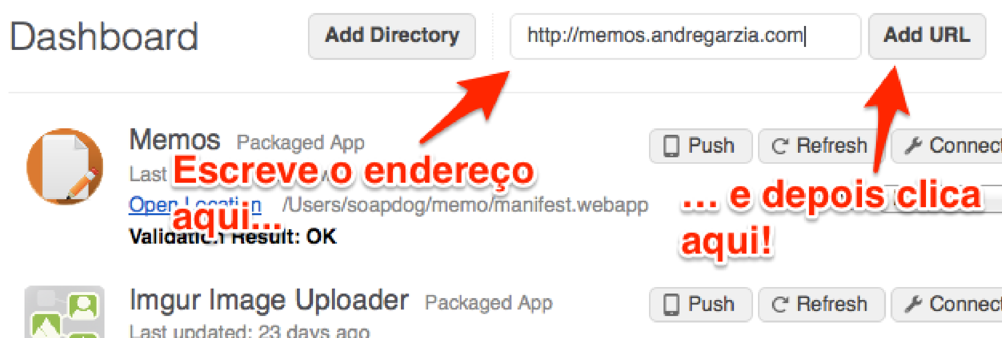
Exemplo de manifesto inválido

Sempre que você fizer alguma alteração no aplicativo você pode apertar o botão **Refresh** para atualizar o app no simulador.

Adicionando aplicativos hospedados

Mesmo que o aplicativo que você pretende lançar como hospedado esteja dentro da mesma máquina que está rodando o simulador, você deve testá-lo utilizando um servidor web. Se você utilizar o procedimento descrito acima para testar um app hospedado pode ser que você tenha algum bug na configuração do seu servidor (tipo servir o manifesto com o MIME Type errado) que você não irá perceber.

Muitos dos aplicativos hospedados não são exclusivos para Firefox OS e sim sites/webapps adaptáveis para o dispositivo que está acessando naquele momento. Esse tipo de site normalmente possui sistemas de backend que você não tem como testar direito com o procedimento de aplicativos empacotados portanto o ideal é testar a partir do servidor web. Para fazer isso você deve escrever o endereço do site ou do manifesto no campo de texto ao lado do botão **Add URL**.



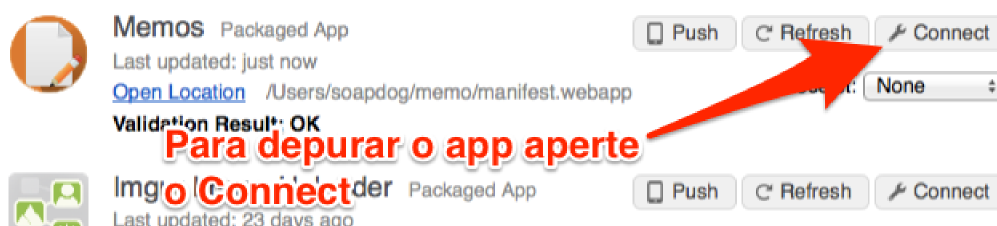
Adicionando um app hospedado ao simulador

Ao clicar no botão, o manifesto é verificado e caso esteja correto o aplicativo é adicionado ao simulador que prontamente é iniciado com o app rodando. Assim como o procedimento para apps empacotados o dashboard mostrará se der algum erro no manifesto.

Sempre que você fizer alguma alteração no aplicativo você pode apertar o botão **Refresh** para atualizar o app no simulador.

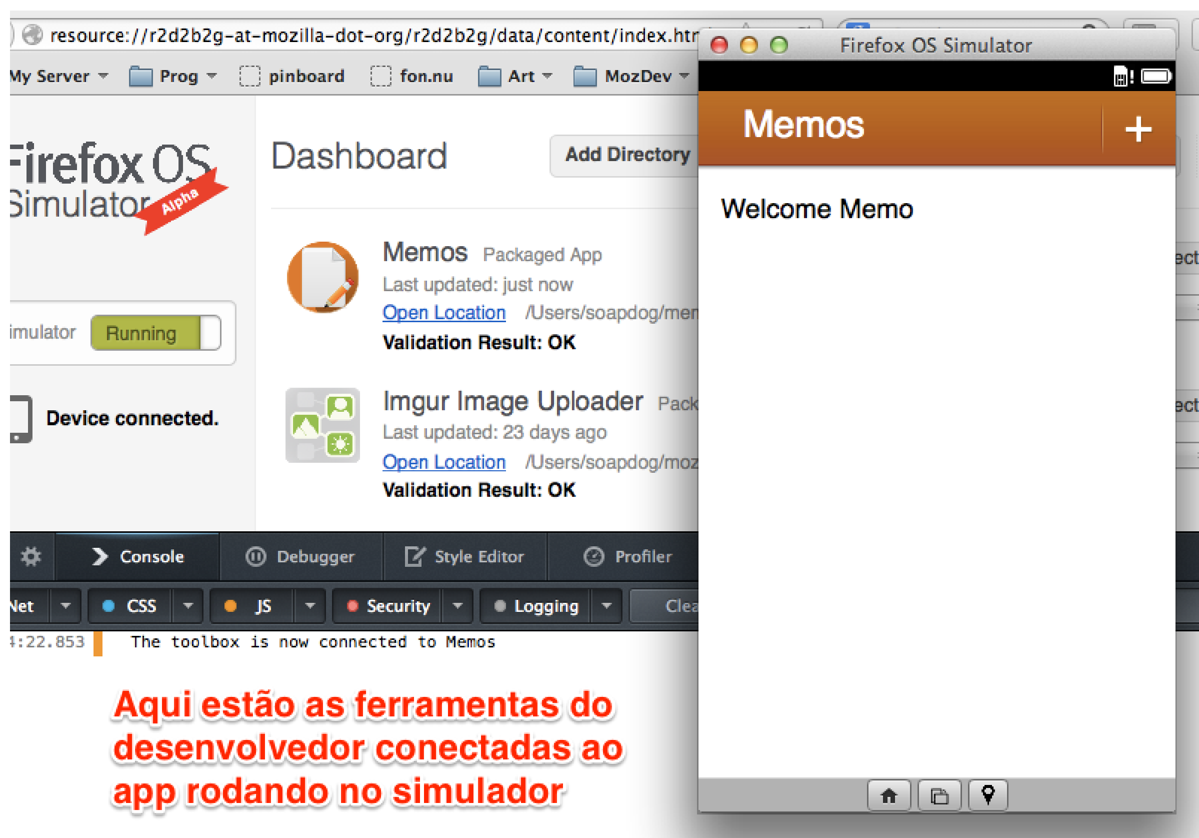
Debugando

Com o aplicativo rodando no simulador já estamos aptos a depurá-lo. Para isso basta clicar no botão **Connect** ao lado da entrada do aplicativo no dashboard do simulador. Isso irá abrir o simulador com o aplicativo escolhido rodando e conectar o **Console de Javascript** no app. Abaixo podemos ver qual botão apertar.



Qual botão apertar para depurar um app

Após apertar esse botão você verá uma tela semelhante a esta:



Ferramentas do desenvolvedor conectadas ao simulador

Com as ferramentas ligadas você pode testar o JavaScript, depurar o DOM, mudar o estilo das coisas, enfim como diz a galera de startup *pivotar até seu app dar certo.*

Após seu app estar rodando bem no simulador é hora de testar em um aparelho de verdade.

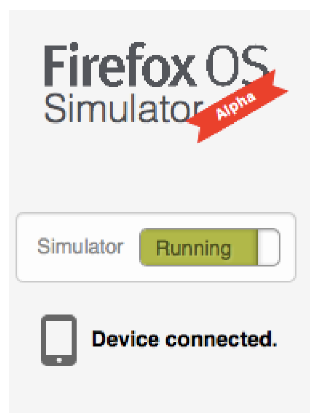
Enviando para o aparelho

Nada substitui testar em um aparelho de verdade. No simulador você fica clicando as coisas com o mouse em uma tela de computador, em um aparelho real você coloca seus dedos em uma tela

de telefone, é uma experiência bem diferente. Só para exemplificar esse tipo de coisa: a alguns anos atrás eu e o Raphael Eckhardt (que fez a capa do livro) estavamos fazendo um jogo tipo puzzle não muito diferente de um bejeweled da vida que envolvia arrastar e soltar peças do jogo no "tabuleiro". Enquanto testavamos no simulador tudo estava uma maravilha porém quando colocamos no telefone notamos que a mão ficava na frente do tabuleiro e que as peças eram tão pequenas que sumiam embaixo do dedo enquanto arrastavamos. Pronto tivemos que mudar nossa arte e nosso UX pois mouses são muito diferentes de mãos.

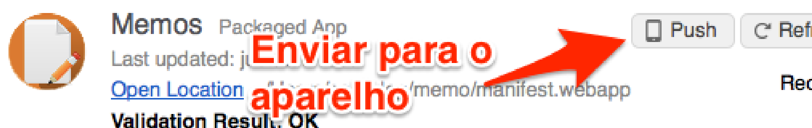
Você pode comprar um aparelho de desenvolvedor rodando Firefox OS na loja da geeksphone[66]. Eu recomendo o Geeksphone Keon[67] pois ele é mais similar aos aparelhos que estão sendo vendidos para os consumidores finais em termos de resolução e capacidade. Você pode também instalar o Firefox OS em alguns aparelhos que rodam Android porém esse tipo de coisa é destinada somente a entusiastas que sabem muito bem o que estão fazendo e possuem mais bugs que as versões que rodam nos Geeksphone e ZTEs.

Caso você possua um aparelho que rode o Firefox OS e esteja com seus drivers em dia você pode enviar aplicativos direto do simulador para o telefone conectado ao computador. Quando o simulador detecta que você plugou um telefone com Firefox OS ele mostra um aviso **Device Connected**.



Aparelho Conectado!

Com o telefone conectado (e detectado), o simulador irá adicionar um botão perto do botão **Connect** e **Refresh** chamado **Push**. Ao apertar esse botão, um pedido de **permissão aparece na tela do telefone** e caso o usuário aceite, o aplicativo é instalado no aparelho.



Qual botão apertar para enviar o app para o aparelho conectado

E a janela de permissão se parece com a foto abaixo.

[66] http://shop.geeksphone.com/en/
[67] http://www.geeksphone.com/



Não é a melhor foto do mundo mas mostra a janela de permissão (são 4:25 AM)

Com o aplicativo rodando no aparelho você pode utilizar *remote debugging* para conectar um console de javascript e depurar o seu app.

Conclusão

A conclusão é o simulador do Firefox OS é legal demais!!! Brincadeiras a parte, a essa altura do livro você ja tem uma boa idéia de como funciona o desenvolvimento de aplicativos para o Firefox OS por isso no próximo capítulo vamos ver como distribuir nosso aplicativos.

Distribuindo Seus Apps

Temos o aplicativo pronto basta saber como fazemos para distribuir o treco. Nesse capítulo veremos como distribuir o seu aplicativo **fora do Firefox Marketplace**. Como vimos na Introdução a Mozilla não tenta impedir sua liberdade de maneira nenhuma portanto podemos distribuir nossas criações como acharmos melhor.

Essa parte de distribuição fora do Firefox Marketplace faz mais sentido (na minha opinião) para aplicativos de interesse segmentado como por exemplo apps para utilização interna em empresas ou somente para os seus clientes. Aplicativos no marketplace estão disponíveis para qualquer um baixar ou comprar. A única maneira de um app que está no marketplace restringir a utilização é com algum sistema de usuários dentro do app (e mantido por um backend) como é o caso de apps como o *Evernote* que pede uma conta do seu próprio sistema para utilização da app. Outro caso onde faz sentido uma distribuição fora do Marketplace é quando você já tem um canal de marketing montado e é capaz de atingir um grande número de pessoas independente do mercadinho. Por exemplo se um site como o *Jovem Nerd* faz um app para Firefox OS, faz sentido distribuir o app no próprio site (além de distribuir também no marketplace).

O processo de distribuição de aplicativos hospedados e empacotados é semelhante porém com chamadas a funções diferentes por isso vou mostrá-los separadamente. Independente se o seu app é hospedado ou empacotado você deve criar um botão ou página que execute a chamada de instalação. Isso pode ser um botão tipo **Instale o Nosso App** ou um endereço especial que quando aberto causa a execução da rotina de instalação. Ao executar as chamadas de instalação, o Firefox OS pergunta para o usuário se ele deseja instalar o app ou não portanto não tem como instalar um app sem a confirmação do mesmo.

Aplicativos hospedados

Rotina para instalação de uma app hospedada

```
var installapp = navigator.mozApps.install(manifestURL);
installapp.onsuccess = function(data) {
  // A App foi instalada
};
installapp.onerror = function() {
 // A App não foi instalada, informações em
 // installapp.error.name
};
```

No exemplo acima `manifestURL` deve conter o endereço do arquivo de manifesto. Ao executar essa rotina o Firefox OS pede para o usuário confirmar a instalação do app e dependendo da ação do mesmo o callback de erro ou de sucesso é chamado.

Para maiores informações sobre essas rotinas veja a página na MDN sobre instalação de aplicativos[68].

Aplicativos empacotados

A instalação de aplicativos empacotados é bem semelhante porém em vez de chamar `mozApps.install()` devemos chamar `mozApps.installPackage()` como podemos ver no exemplo abaixo.

Rotina para instalação de uma app empacotado

```
var packageURL = "http://myapp.example.com/myapp.zip"; // <-- absolute url to \
package
var installapp = navigator.mozApps.installPackage(packageURL);
```



Perigo: Eu tenho a impressão que a instalação de apps empacotados fora do marketplace ainda não é possível na versão atual do Firefox OS. Por mais que a API esteja documentada acho que ela ainda não foi totalmente implementada. Eu nunca testei. Aviso dado. Se funcionar me manda email para eu atualizar o livro.

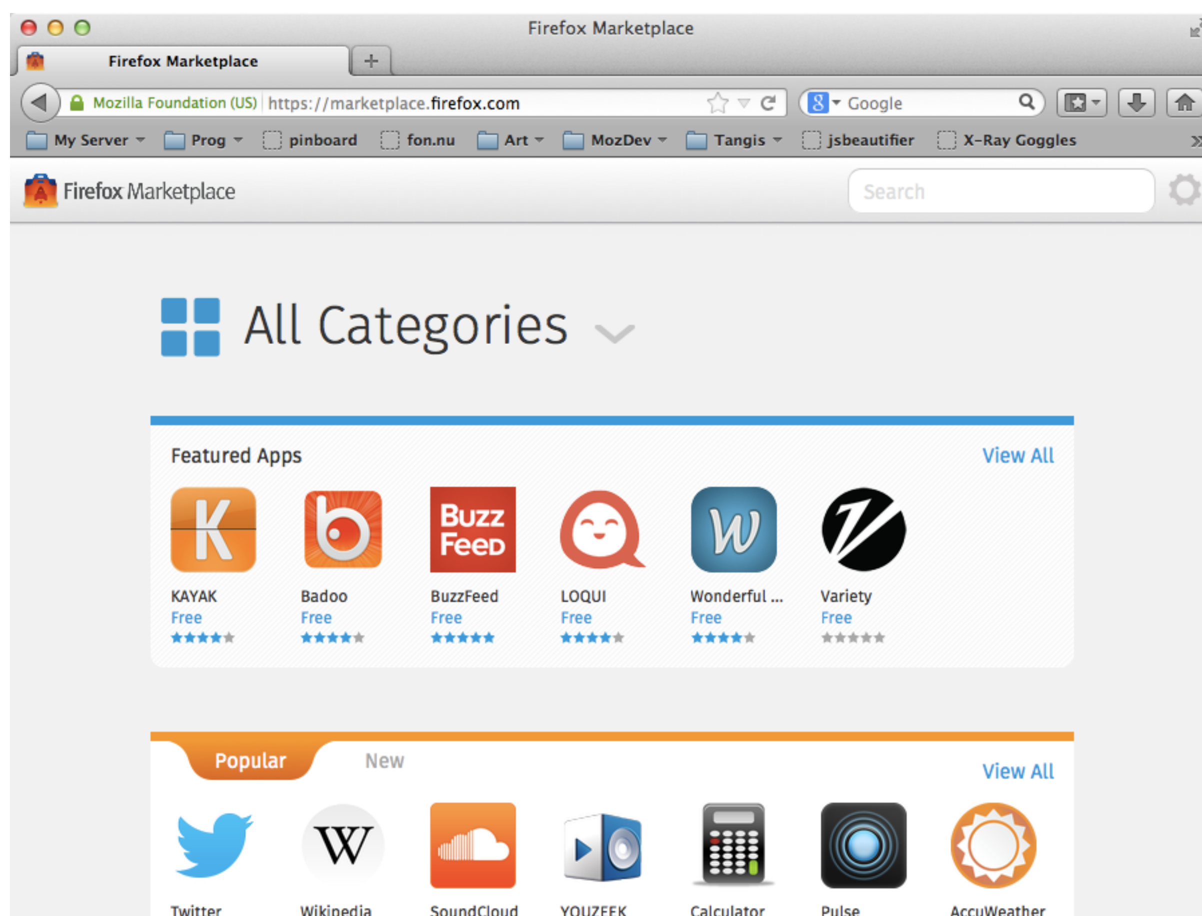
Conclusão

Vimos rapidamente como é realizada a instalação de um aplicativo fora do Firefox Marketplace atraves das rotinas de instalação e gerenciamento e de **Open Web Apps**. Existem várias outras rotinas como por exemplo para detectar se a app está instalada ou não (assim você pode esconder o botão de instalar se a app já está instalada). Para saber mais sobre essas rotinas não deixe de ver a página na MDN sobre instalação de aplicativos[69] (eu já falei isso nesse capítulo, eu sei sou prolixo).

No próximo capítulo vamos ver como distribuir seu app no Firefox Marketplace.

[68] https://developer.mozilla.org/pt-BR/docs/Apps/JavaScript_API
[69] https://developer.mozilla.org/pt-BR/docs/Apps/JavaScript_API

O Firefox Marketplace



Firefox Marketplace

O Firefox Marketplace[70] é o mercadinho do Firefox onde você pode pegar ou comprar apps para o Firefox OS, Firefox Desktop e Firefox para Android. Ele é o principal canal para distribuição de aplicativos para Firefox OS porém você não precisa usá-lo se não quiser.

Para distribuir apps no marketplace você precisa estar identificado via Mozilla Persona[71]. Basta clicar **Sign Up** e seguir as instruções. Uma vez identificado, você esta pronto para enviar apps para o Firefox Marketplace.

Checklist antes de pensar em enviar o seu app

Todo aplicativo que é enviado para o marketplace passa por um processo de aprovação. Aplicativos normais passam por um processo menos rigoroso que aplicativos privilegiados pois

[70] http://marketplace.firefox.com
[71] https://login.persona.org/about

utilizam APIs menos sensíveis. Antes de enviar o aplicativo familiarize-se com os critérios de avaliação do marketplace[72]. Os pontos mais importantes são:

- O Firefox OS não tem um botão de voltar como o navegador ou o Android. Se dentro da sua app o usuário navegar para algum ponto e não tiver como voltar mais (ficar preso em uma tela), seu app será rejeitado.
- Seu app deve conter os ícones 60x60 utilizados pelo Firefox OS e uma descrição detalhada do que ele faz no manifesto.
- Seu aplicativo deve fazer o que diz a descrição dele. Você não pode falar que o seu app faz uma coisa que ele não faz.
- Se o seu app pede uma permissão para alguma funcionalidade ele deve usá-la. Marcar o app como privilegiado e não utilizar nada das APIs privilegiadas fará com que o seu app seja rejeitado com um pedido para alterar o tipo para normal.
- Seu app precisa ter uma política de privacidade bem explicada no marketplace.
- O manifesto e o aplicativo devem vir do mesmo domínio no caso de apps hospedados.

Existem outros critérios como mostrado nos links acima. Vale a pena estudar esses critérios antes de enviar o app afinal perde-se muito tempo tendo o aplicativo rejeitado por uma bobagem que você resolve em cinco minutos.

Preparando seu app para envio

A preparação do seu app para envio depende do tipo de app que ele é. Aplicativos hospedados simplesmente precisam estar disponíveis em algum servidor web. Aplicativos empacotados devem ser zipados antes do envio e merecem mais atenção.

Muita gente comete o erro de selecionar a pasta que contém os arquivos do aplicativo e mandar zipar. Isso faz com que o zip contenha uma pasta e essa pasta contenha todos os arquivos. Esse não é o jeito correto de zipar apps para Firefox OS. O correto é fazer com que o zip contenha diretamente os arquivos do aplicativo. Mais especificamente, o que é necessário é que o manifesto esteja na *raiz do arquivo zip*, ou seja, que ele não esteja dentro de nenhuma pasta dentro do arquivo compactado. No Mac OS X e no Linux podemos navegar no terminal até dentro da pasta que está o nosso aplicativo e executar um comando como `zip -r meuapp.zip *` como podemos ver na captura de tela a seguir.

[72]https://developer.mozilla.org/en-US/docs/Web/Apps/Publishing/Marketplace_review_criteria

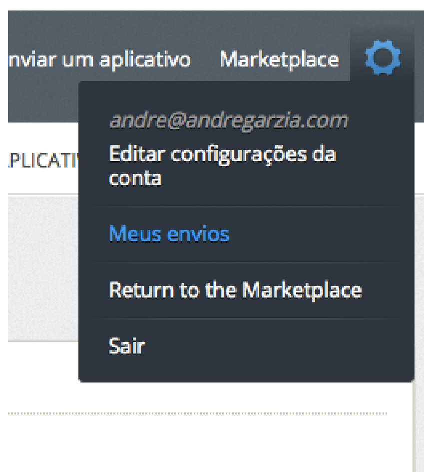
```
ndres-MacBook-Air:firefoxos-sample-app-image-uploader soapdog$ zip -r imgur.zip *
  adding: LICENSE (deflated 67%)
  adding: README.md (deflated 58%)
  adding: app.js (deflated 69%)
  adding: building_blocks/ (stored 0%)
  adding: building_blocks/building_blocks.css (deflated 84%)
  adding: building_blocks/cross_browser.css (deflated 78%)
  adding: building_blocks/fonts/ (stored 0%)
  adding: building_blocks/fonts/FeuraSans-1.0/ (stored 0%)
  adding: building_blocks/fonts/FeuraSans-1.0/FeuraSans-Bold.woff (deflated 0%)
  adding: building_blocks/fonts/FeuraSans-1.0/FeuraSans-Light.woff (deflated 0%)
  adding: building_blocks/fonts/FeuraSans-1.0/FeuraSans-Medium.woff (deflated 0%)
  adding: building_blocks/fonts/FeuraSans-1.0/FeuraSans-Regular.woff (deflated 0%)
  adding: building_blocks/fonts/FeuraSans-1.0/LICENSE (deflated 36%)
  adding: building_blocks/fonts.css (deflated 85%)
  adding: building_blocks/icons/ (stored 0%)
```

Zipando corretamente os arquivos

O arquivo zip final é o que enviaremos para o marketplace.

Envio para o Firefox Marketplace

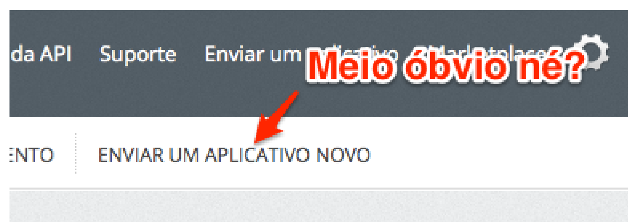
De posse do seu app pronto e da certeza que ele preenche todos os requisitos para ser aprovado, você deve navegar para **Meus Envios**[73] utilizando o botão da engrenagem.



Meus Envios

Dentro da página de gerência dos seus aplicativos, você deve clicar em **Enviar um novo aplicativo** no menu superior.

[73]Em português *My Submissions*.



Link para enviar novo aplicativo

Esse link o levará para o formulário de entrada de novo aplicativo que pode ser visto na tela abaixo.

Enviar novo app

Nessa tela você pode selecionar as seguintes opções:

- Se o aplicativo é hospedado ou empacotado.
- Se ele é gratuíto ou pago (ou com *in-app purchases*).
- Quais aparelhos ele funciona (Firefox OS, Firefox Desktop, Firefox for Mobile no Telefone, Firefox for Mobile no Tablet).

Feitas essas escolhas você passa para a segunda tela. Nesse livro estamos fazendo o processo para o envio de um app empacotado porém os demais tipos de app tem um workflow semelhante.

Estamos assumindo para efeito desse exemplo um aplicativo empacotado gratuito para Firefox OS. Nesse caso, ele pede que a gente faça upload to arquivo que preparamos na etapa anterior.

Após o envio do arquivo ele será processado e um relatório com mais opções será mostrado.

Depois do envio do arquivo zip

Como podemos ver na captura, esse programa que eu enviei não contém erros mas contém seis avisos. Fora isso podemos marcar os **requisitos mínimos** para que o app seja instalado. No caso desse exemplo, o último requisito, que fala sobre resolução de tela qHD deve ser desmarcado visto que esse aplicativo funciona em qualquer resolução.

O próximo passo chamado **Passo #3: Detalhes** é onde você preenche as informações sobre o seu aplicativo tais como categoria, descrição, fornece capturas de tela entre outros.

Enviar um aplicativo
1 Acordo
Editar detalhes do aplicativo
Por favor, revise suas informações listadas abaixo. Nós já preenchemos algumas informações obtidas do seu mar
Imgur Image Uploader
https://marketplace.firefox.com/.../ imgur-image-uploader Abrir a página de detalhes
Editar
Description: *
An app to share images using the imgur service
Essa descrição aparecerá na página de detalhes.
Categorias: * Escolha uma ou duas categorias.
Entretenimento e esportes
Negócios
Jogos
Música
Notícias e clima
Produtividade
Social e comunicação
Viagem
Livros e referências
Educação
Saúde e fitness
Estilo de vida
Fotos e mídia
Utilitários
Compras

Preenchendo detalhes

Após preencher os detalhes o processo de envio está concluído e ai basta aguardar a aprovação dos revisores do marketplace. Parabéns você tem um aplicativo no Firefox OS!!!

Na página de Gerência de Aplicativos[74] você pode verificar o status dos seus envios e alterar detalhes se necessário.

Para saber mais sobre o envio de aplicativos para o Firefox Marketplace leia esse artigo na central do desenvolvedor do Firefox OS[75].

Conclusão

Parabéééééns!!!! Você agora tem um aplicativo no Firefox Marketplace, você esta desbravando um mercado todo novo!

Espero que tenha gostado desse guia rápido. Eu pretendo atualizar esse guia constantemente (inclusive achando um bom editor para consertar meu analfabetismo) portanto fique atento para novidades. Se você baixou esse livro do Leanpub então fique tranquilo que você será avisado das atualizações.

[74] https://marketplace.firefox.com/developers/submissions
[75] https://marketplace.firefox.com/developers/docs/submission

Não deixe de me dar algum feedback. Eu fiquei a noite inteira acordado escrevendo esse livro, ou seja, eu realmente gosto desse projeto! Eu fico constantemente olhando o Twitter onde minha conta é @soapdog[76].

Agora que você faz parte dos criadores de aplicativos para Firefox OS venha fazer parte da Comunidade Mozilla Brasil[77] e ajudar projetos livres como o Firefox OS a crescer ainda mais.

[76] http://twitter.com/soapdog
[77] http://mozillabrasil.org.br

Apêndice 1: Links úteis

- Mozilla[78]
- Comunidade Mozilla Brasil[79]
- Central do Desenvolvedor do Firefox OS[80]
- Mozilla Developers Network[81]: Melhor documentação ever!
- Firefox OS[82]
- Wiki da WebAPI[83]

[78] http://mozilla.org
[79] http://mozillabrasi.org.br
[80] http://marketplace.firefox.com/developers
[81] http://developer.mozilla.org/
[82] http://www.mozilla.org/pt-BR/firefox/os/
[83] http://wiki.mozilla.org/WebAPI